Proyecto Final de Carrera - Ingeniería Informática - Curso 2009/2010

Implementación de una aplicación web para el cálculo de la infiltración acuífera bajo los paradigmas de reutilización de componentes e interoperabilidad de sistemas

Autor: Angel Armingol García

Director: Raquel Miguel Sauco

Ponente: Pedro Muro Medrano

Dep. de Informática e Ingeniería de Sistemas (DIIS)
Centro Politécnico Superior
María de Luna, 1
50018 Zaragoza (España)

GeoSpatiumLab S.L.
Carlos Marx, 6
50015 Zaragoza (España)

Septiembre 2010

Implementación de una aplicación web para el cálculo de la infiltración acuífera bajo los paradigmas de reutilización de componentes e interoperabilidad de sistemas

RESUMEN

Actualmente el uso eficiente y responsable de los recursos hídricos se ha convertido en una necesidad. Para ello el estudio de las características hidrólogicas es fundamental, y uno de los parámetros que se puede calcular de un recinto o lugar, es la cantidad de agua que absorve el terreno en un instante determinado, denominada infiltración acuífera. En este contexto surge una colaboración dentro de GeoSpatiumLab S.L. con una empresa del sector medio ambiental para realizar una aplicación web que automatice el tedioso proceso del cálculo de la infiltración acuífera.

La aplicación web desarrollada es capaz de obtener los datos de entrada necesarios para el cálculo de la infiltración a través de la interfaz, manipularlos y transformarlos para ofrecer al usuario final los resultados del mismo, a través de la propia interfaz web, representados en informes o gráficas.

La construcción de la aplicación web final que realiza el cálculo de la infiltración se ha nutrido con componentes nuevos desarrollados durante el proyecto, como la librería de generación de informes y la librería de generación de gráficas, y con otros existentes, algunos de los cuales, han sido ampliados o mejorados para adaptarse a la necesidades, manteniendo su genericidad. De esta manera, gracias a la reutilización de componentes, se ha podido minimizar el tiempo de desarrollo.

Para el desarrollo de todas estas tareas se ha utilizado lenguaje de programación Java, apoyándose en el framework GWT (Google Web Toolkit) en el caso de la aplicaciones web. Para el aspecto visual de estas últimas se han utilizado hojas de estilo CSS (Cascade Style Sheets). Como herramienta para la gestión y construcción de proyectos se ha hecho uso de Maven y como gestor de versiones se ha utilizado GIT en conjunto con Subversion.

A la memoria de mi abuelo José Armingol.

Agradecimientos

En primer lugar, me gustaría dar las gracias a mi directora, Raquel Miguel Sauco por su interés y dedicación en todo momento. Por otro lado, quisiera destacar el buen ambiente y el trato recibido por todos los miembros de GeoSpatiumLab, y en particular agradecer a Íñigo Galaz sus valiosas explicaciones y su paciencia.

Quiero agradecer a mis padres, Ángel y Carmen, su apoyo incondicional durante toda mi vida y en particular en los momentos más duros de la carrera. Sin olvidar a mi hermana Tamara que pone siempre el toque de felicidad en mi día a día. A mi tía Marimar y a mis abuelos Carmen y Emilio por su interés y cariño.

Y como no, a muchos de los compañeros y amigos de carrera con los que he compartido varios años.

En definitiva, muchas gracias a todos los que han hecho posible que hoy esté aquí.

Índice general

Capítulo 1

Introducción

1.1. Objetivos y justificación del proyecto

La infiltración acuífera se define como el proceso por el cual el agua penetra desde la superficie del terreno hacia el suelo pasando a formar parte del agua subterránea. Para calcular el valor de la infiltración acuífera en un instante determinado, uno de los métodos es del denominado "método del número de curva". Éste consiste en la realización de múltiples operaciones encadenadas (ver anexo E) que, aunque son sencillas, se realizan sobre gran cantidad de datos.

GeoSpatiumLab dispone de componentes genéricos que podían servir de base para la construcción de una aplicación web para realizar el cálculo de la infiltración minimizando, por tanto, el coste de desarrollo del mismo.

Así pues, el objetivo de este proyecto es la realización de una aplicación web que permita automatizar la captura de los datos de entrada necesarios (precipitaciones diarias, latutides, temperaturas...) y que permita realizar el tedioso cúmulo de operaciones necesarias para calcular la infiltración acuífera, presentándo los resultados de una manera ágil y fácilmente analizable. Todo ello bajo los paradigmas de la reutilización y la interoperabilidad.

1.2. Descripción del proyecto

El proyecto consiste en diseñar e implementar varios componentes genéricos, reutilizar otros existentes y, todos ellos, hacerlos interoperar en una aplicación web que permita, dado unos datos de entrada, calcular la infiltración de uno o varios recintos. Además el usuario será capaz de visualizar los resultados a través de informes y gráficas que faciliten su análisis.

Más en detalle, las características del proyecto son:

- Mejora de la aplicación web de gestión de datos existente. Se ha modificado la aplicación web genérica de tratamiento de datos de la que dispone la empresa para que pueda trabajar adecuadamente con los datos del cálculo de la

infiltración, ya que éstos, por su naturaleza, pueden contener series muy largas y producir desbordamientos de memoria. Adicionalmente, se ha buscado una solución para facilitar al usuario el manejo ágil de estas series.

- Creación de una librería de generación de informes. Se ha diseñado e implementado de una librería de generación de informes fácilmente configurable que permite incluir texto, imágenes y mapas provenientes de cualquier servidor de mapas OGC estándar. Para ello, primero se han estudiado las distintas alternativas existentes de motores de creación de informes del mercado.

- Creación de una librería de generación de gráficas. Se ha implementado una librería genérica capaz de elaborar gráficas, basada en un servicio web existente que ofrece esta funcionalidad.

- Implementación de la aplicación final. Se creará la aplicación web para el cálculo de la infiltración acuífera integrando los diversos componentes desarrollados o mejorados durante el proyecto.

1.3. Entorno de trabajo

Este proyecto ha sido realizado en GeoSpatiumLab[1], empresa spin-off de la Universidad de Zaragoza y, en concreto, del Grupo de Sistemas de Información Avanzados (IAAA)[2].

GeoSpatiumLab es una empresa de base tecnológica, Spin-off de la Universidad de Zaragoza, especializada en el tratamiento digital de la información geoespacial y georreferenciada y sus ámbitos de aplicación. Uno de sus campos de actividad es el medioambiente, en el que trabaja en el desarrollo de tecnologías que permitan optimizar los procesos de obtención y explotación de los datos necesarios para la gestión de los recursos hídricos, la prevención de riesgos naturales, la elaboración de planes de acción y la gestión de catástrofes naturales o la definición y gestión de los espacios naturales protegidos, entre otras muchas actividades. Es en este campo (el del medioambiente), donde se enmarca el presente proyecto final de carrera.

GeoSpatiumLab aplica habitualmente conceptos tan importantes en la ingeniería del software como son la reutilización y la interoperabilidad, permitiendo reducir costes, tiempo de desarrollo e incrementando la productividad, conceptos que han sido aplicados a largo de todo el proyecto.

1.4. Estructura de la memoria

Este documento se ha divido en dos partes, la primera es el núcleo de la memoria y pretende dar una impresión global del trabajo realizado de forma concisa y sin

[1] http://www.geoslab.es
[2] http://iaaa.cps.unizar.es

introducir detalles de bajo nivel. La segunda parte, compuesta por diferentes anexos pretende mostrar una descripción más concreta y profunda de la labor realizada, tanto de los resultados finales, como del proceso seguido para obtenerlos. En detalle los capítulos y anexos son:

El capítulo 1, al cual pertenece esta sección, describe los objetivos y alcance del proyecto.

Capítulo 2: Explicación del trabajo realizado.

Capítulo 3: Presenta las conclusiones del proyecto y el trabajo posterior que se podría desarrollar.

Capítulo 4: Lista de acrónimos usados a lo largo de la memoria.

Anexo A: Presenta la planificación, el control de esfuerzos y métricas.

Anexo B: Explica la metodología de de trabajo usada a lo largo de todo el proyecto.

Anexo C: Análisis y diseños en detalle de los diferentes componentes.

Anexo D: Manual de usuario de la aplicación y de las librerías desarrolladas.

Anexo E: Explicación de las operaciones necesarias para llevar a cabo el cálculo de la infiltración acuífera.

Anexo F: Es el último capítulo de la memoria y presenta la definición y explicación de términos.

Capítulo 2

Descripción del trabajo realizado

En este capítulo se desglosa el trabajo realizado. Como introducción se expone la situación de partida donde se detallan los componentes que ya existían en la empresa al inicio de este proyecto, posteriormente se muestra la arquitectura de alto nivel de la aplicación y, por último, en las secciones siguientes se describen las modificaciones realizadas en componentes ya existentes, así como los nuevos desarrollados durante el proyecto. Por último se describe la aplicación web final.

2.1. Situación inicial

GeoSpatiumLab dispone de componentes genéricos que podían servir de base para la construcción de una aplicación web para realizar el cálculo de la infiltración.

Para las cuatro funcionalidades básicas que debía ofrecer la aplicación, se han reutilizado, previa ampliación o mejora en algunos casos, los siguientes componentes:

1. **Captura de datos:** Para la captura de datos se podía hacer uso de *WebFTManager* que es una aplicación web para el tratamiento de datos genéricos. (ver sección 2.7).

2. **Procesamiento:** Para la realización del cálculo de la infiltración acuífera, GeoSpatiumLab dispone de un servicio web un servicio de resolución de polinomios y a un servicio web de resolución de tablas de decisión, que poseían las funcionalidades requeridas.

3. **Almacenamiento:** Para el almacenamiento en base de datos se podía, al igual que en el caso de la captura de datos, hacer uso de *WebFTManager*, ya que es capaz de almacenar los datos con los que trabaja.

4. **Presentación:** Para la visualización de los resultados, *WebFTManager* podía mostrar los resultados en la propia interfaz, para representar los resultados con gráficas se ha creado una librería a partir de un servicio web *(WebChartService)* del que se ha aprovechado su funcionalidad para desarrollar la librería *GSL-Charts* (ver sección 2.4), en el caso de los informes sí que se ha desarrollado una librería (*GSL-Reports*) desde cero (ver sección 2.3).

En las secciones siguientes se detalla en qué medida estos componentes han tenido que ser modificados para conseguir que interoperaran correctamente para formar la aplicación web resultante *InfiltrationSolver* (ver sección 2.8).

2.2. Arquitectura de alto nivel

La aplicación web final (*InfiltrationSolver*) es un caso particular de la aplicación web genérica *WebFTManager* de la que ya disponía la empresa. *InfiltrationSolver* se aprovecha de la funcionalidad ya disponible de *WebFTManager* y personaliza las características que se requieren para el caso específico de este proyecto (estructura de datos, aspecto visual de la aplicación, mecanismos para calcular la infiltración...)

El diseño de la arquitectura puede verse en la figura 2.1.

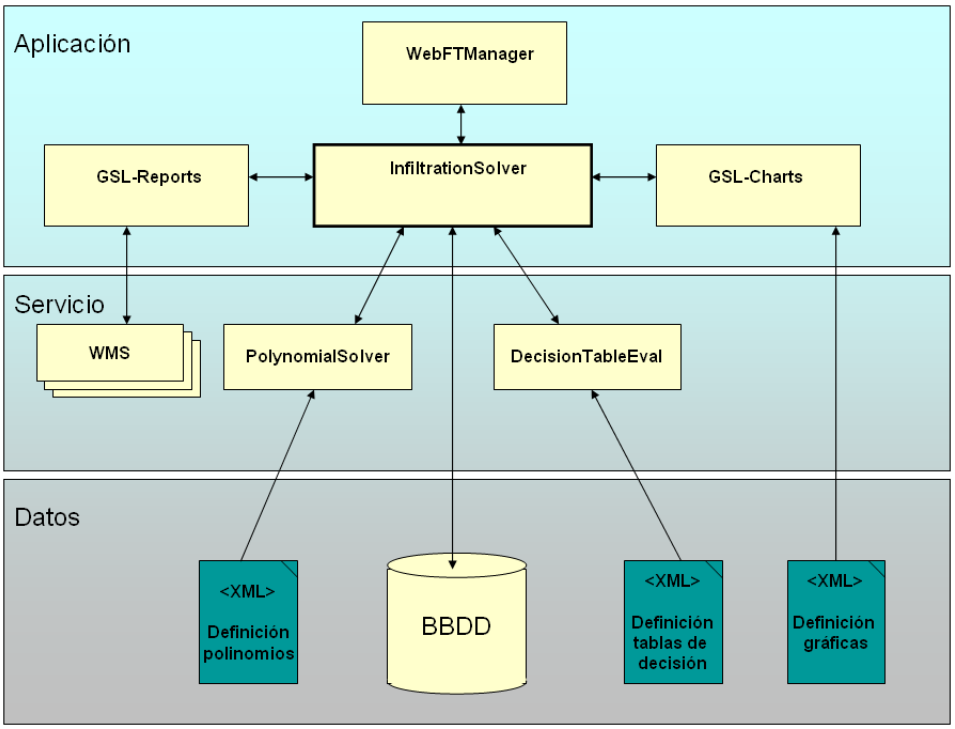

Figura 2.1: Arquitectura de alto nivel

En el nivel de aplicación, el más cercano al usuario final, se encuentra *InfiltrationSolver* que será la aplicación web con la que el usuario podrá realizar el cálculo de la infiltración. En este nivel también se encuentran el resto de componentes genéricos que forman parte de *InfiltrationSolver: WebFTManager* como aplicación genérica de tratamiento de los datos, *GSL-Reports* como librería de generación de informes, el cuál se comunica a su vez con los servidores de mapas (WMS ver sección F.4) para obtener los mapas que deba incluir en los informes, y *GSL-Charts* como librería de generación de gráficas.

Para realizar el cálculo de la infiltración acuífera se necesitan realizar una serie de operaciones (ver anexo E). Para su resolución, InfiltrationSolver se comunica con los

servicios de resolución de polinomios (*PolynomialSolver*) y de resolución de tablas de decisión (*DecisionTableEval*).

2.3. GSL-Reports

GSL-Reports es una librería para la generación de informes genérica y por tanto, puede utilizarse como componente en cualquier aplicación Java que lo necesite.

Aunque la librería está desarrollada con un motor en concreto (BIRT[1] en este caso) el diseño y la implementación están pensados para dejar la posiblidad de poder extender la librería con otros motores diferentes. La elección de BIRT, viene dada ya que previamente se había trabajado en la empresa con este motor satisfactoriamente. No obstante, se verificó que las nuevas versiones satisfacían los requeremientos de la librería.

A modo de resumen, la librería desarrollada es capaz de:

- Generar informes en diferentes tipos de datos (PDF, DOC)
- Incorporar imágenes en una posición del informe concreta o al final del documento tanto especificando una ruta remota (URL o local) como con un flujo de bytes (InputStream).
- Incorporar mapas de uno o varios servidores de mapas pudiendo definir SLD.
- Adición y limpieza de filtros para adaptarse a posibles restricciones de los datos.
- Cambiar la fuente de donde BIRT obtiene los datos de los informes.

En la figura 2.2 se puede ver un ejemplo de informe generado con la librería.

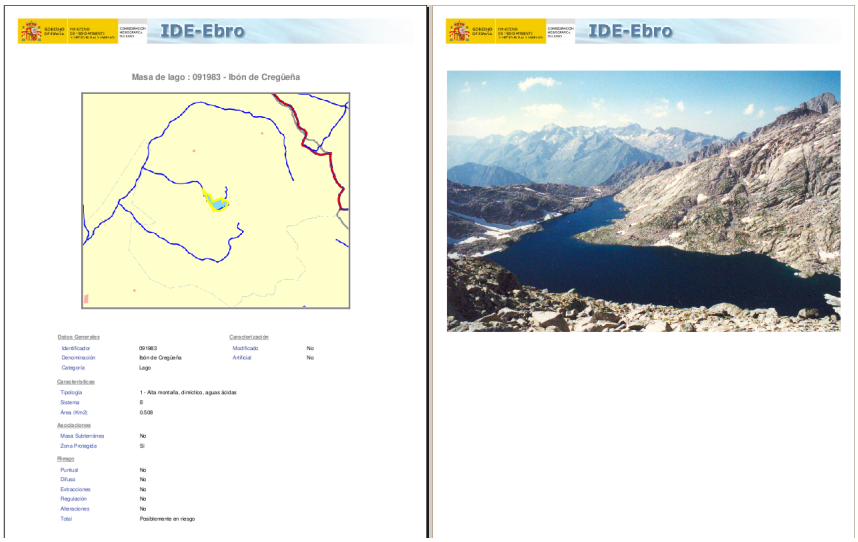

Figura 2.2: Ejemplo de informe generado con GSL-Reports

[1] http://www.eclipse.org/birt/phoenix/

2.3.1. Arquitectura

En la figura 2.3 podemos apreciar la arquitectura de la librería de generación de informes (*GSL-Reports*). La aplicación de usuario que haga uso de esta librería debe proveerla de una plantilla de informe prediseñada. Una vez que la librería dispone de la plantilla ya se puede personalizar siguiendo los pasos que se destacan en el manual de usuario (sección D.2).

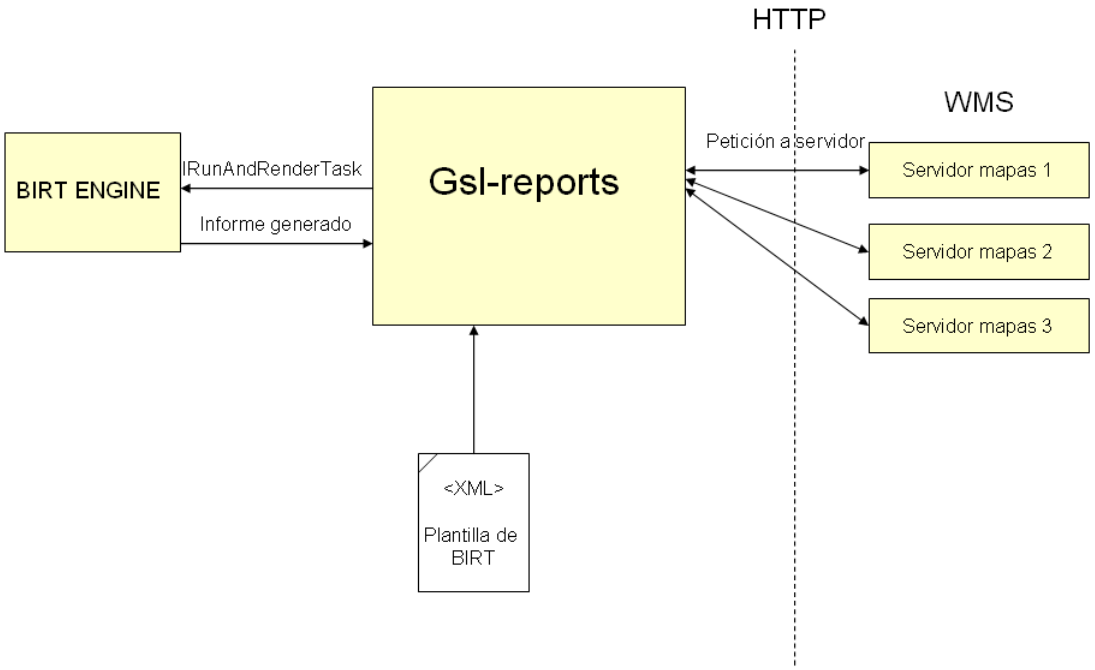

Figura 2.3: Arquitectura de la librería de generación de informes

La librería es capaz de procesar diferentes peticiones a diferentes servidores de mapas y, posteriormente, fusionar las imágenes provenientes de los mismos en una única imagen que se incluirá en el informe generado. Existen diferentes peculiaridades respecto a las peticiones de mapas que se explican en la sección 2.3.2.

Los diseños en detalle de *GSL-Reports* pueden verse en la sección C.1.

2.3.2. Trabajo realizado

El trabajo realizado en esta librería se ve reflejado en las siguientes características.

Generar el informe en diferentes formatos

La librería se ha desarrollado con el objetivo de obtener como resultado de un informe en varios tipos de fichero, tales como PDF o DOC.

Se ha dado soporte tanto al guardado del informe generado en fichero, como al envío del mismo (sea cual sea su formato) como una ristra de bytes (OutputStream). Esta característica es muy utilizada ya que si la librería trabaja en una aplicación web en el lado del servidor, evita la creación de ficheros innecesarios al permitir al cliente obtener la ristra de bytes directamente.

Incorporación de imágenes

GSL-Reports permite la inclusión de imágenes en los informes por ruta remota (URL o local) y por una ristra de bytes (InputStream).

También es posible asociar una imagen a un sitio concreto en nuestro informe o simplemente incluirla al final del documento.

Incorporación de mapas

GSL-Reports permite, del mismo modo que con la imágenes, la incorporación de mapas. De hecho un mapa es un caso particular de imagen. Los mapas se obtienen realizando peticiones a un servidor de mapas (WMS) y al igual que en el caso de las imágenes, podemos añadir estos mapas a nuestro informe en una posición en concreto o al final del documento.

Como clasificación de las características de la funcionalidad de la incorporación de mapas podemos destacar:

- Obtención de un mapa de uno o varios WMS.

  GSL-Reports nos permite incluir un mapa que se construya en base a una o varias peticiones a uno o varios servidores de mapas con una o varias capas por servidor.

  Cuando se hacen peticiones a diferentes servidores de mapas las imágenes obtenidas por cada una de esas peticiones, se superponen unas con otras para obtener la imagen final. De tal modo que la primera petición formará la "base de la imagen", la siguiente se superpondrá y fusionará con ésta para formar una nueva "base de la imagen" repitiendo el proceso con la siguiente petición hasta no haber más peticiones. Así se consigue una única imagen que será la que se incluya en el informe.

  Para aumentar la rapidez en la obtención de los mapas, para cada una de las peticiones se crean diferentes hilos de ejecución y se lanzan en paralelo. Como las respuestas no tienen por qué respetar el orden en que fueron enviadas, previamente se almacena el orden de envió para reordenarlas.

- Ajuste automático del tamaño de imagen

  Otra característica interesante que permite la librería es la adaptación en tiempo de ejecución del BBOX, (el bounding box es un conjunto de cuatro puntos que define un rectángulo que corresponde con la región de mapa a pedir a un WMS) esto significa que si tenemos un bounding box que sabemos corresponde

a una petición de 750x500 píxeles (ratio de aspecto 1.5:1) pero nosotros queremos la imagen en 600x600 píxeles (ratio de aspecto 1:1), la propia librería transformará el BBOX introducido para evitar la deformación de la imagen.

La adaptación del BBOX es opcional ya que el usuario puede no estar interesado en que se autoajuste el bounding box en caso de realizar peticiones a servidores que tengan diferente SRS, ya que el SRS establece el sistema de referencia espacial en base al cual se especificarán las coordenadas de los diferentes parámetros y condiciona el ratio de aspecto de la imagen, ya que la representación de un mapa en un plano depende de éste.

- SLD: Styled Layered Descriptor

  Si el servidor de mapas lo permite, la librería ofrece la posibilidad de realizar peticiones con SLD donde se puede personalizar el mapa resultante de una manera muy flexible. En este caso con la dirección del servicio de mapas y el fichero SLD sería suficiente para realizar la petición.

Cambiar el conjunto de datos o el conjunto de fuentes existente en el informe

El conjunto de datos son las tablas donde se encuentran los datos a mostrar en los informes.

El conjunto de fuentes se define como el lugar de donde se obtiene el conjunto de los datos, por ejemplo una base de datos.

Cambiando el conjunto de fuentes, podemos alterar la base de datos de donde la librería obtiene el conjunto de datos. Ej: Una base de datos ORACLE y otra POSTGRESQL, cambiando el conjunto de datos podemos alterar los datos que se mostrarán en el informe.

Ambas funcionalidades, ofrecen flexibilidad al usuario, pero hay que tener especial cuidado en asegurarse que el conjunto de datos nuevo tenga una estructura idéntica al conjunto de datos anterior para que el informe sepa conseguir los datos.

Adición y limpieza de filtros

Se pueden incorporar filtros al conjunto de datos en tiempo de ejecución, de esta manera se pueden filtrar los datos a mostrar en los informes y es una manera muy sencilla de generar informes específicos (entre dos fechas por ejemplo). Al igual que se pueden añadir, se pueden eliminar.

Paso de parámetros

Un parámetro es una manera de personalizar los datos que se van a mostrar, para generar distintos informes dependiendo de dicho parámetro. Ej: Mostrar ventas de coches, en función del modelo de coche. El mismo informe serviría para mostrar

las ventas de cualquier coche, pero el parámetro (modelo de coche) personaliza el informe.

Una diferencia importante entre los parámetros y los filtros descritos en la sección anterior, es que los filtros se aplican una vez que el motor tiene los datos y los parámetros obtienen los datos ya filtrados delegando la tarea de filtrado a la base de datos. Además los filtros están pensados para añadirse y eliminarse en tiempo de ejecución mientras que los parámetros se crean en la fase de diseño de la plantilla.

Además de para personalizar los datos, se puede utilizar para enviar al informe un dato concreto, como el nombre del informe, el nombre de una columna, etc...

2.4. GSL-Charts

GSL-Charts es una librería de generación de gráficas. Esta librería ha sido construida tomando como base un servicio web existente en la empresa que ya ofrecía esta funcionalidad (*WebChartService*) y que se apoya en el programa estadístico R-Project[2]. *WebChartService* es un servicio web capaz de generar gráficas en base a una plantilla XML, entre sus características están la creación de gráficas de puntos, líneas, sectores...

La librería es capaz de generar gráficas de dos maneras diferentes:

- Incluyendo en el fichero XML tanto los datos como el formato de gráfica
- Incluyendo en el fichero XML sólo el formato e insertando los datos "al vuelo".

2.4.1. Arquitectura

En la figura 2.4 podemos apreciar la arquitectura de la librería de generación de gráficas (*GSL-Charts*). *GSL-Charts* recibe como entradas los datos de una gráfica (opcionalmente) y una plantilla de gráfica Transforma las entradas en datos que entiende el motor de *R-Project* y recoge los resultados.

Los diseños en detalle de *GSL-Charts* pueden verse en la sección C.2.

2.4.2. Trabajo realizado

La construcción de esta librería permite que no haga falta definir los datos de una gráfica en un fichero XML sino que puedan ser asignados en tiempo de ejecución.

El formato de la gráfica (ver sección D.3.4) está prediseñado aunque se podría modificar "al vuelo" ya que este diseño está hecho con XML. Esta optimización se ha rechazado por falta de tiempo aunque si que figura en las posibles mejoras que se detallan en el apartado 3.2.

El trabajo realizado en detalle ha sido:

[2]http://www.r-project.org

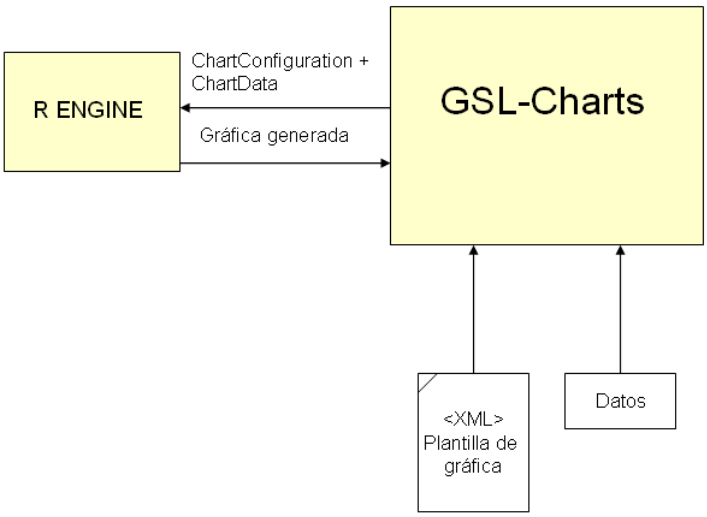

Figura 2.4: Arquitectura de *GSL-Charts*

Convertir *WebChartService* en una librería

Aunque se podría haber aprovechado *WebChartService* como tal y enviar peticiones de generación de gráficas, una librería ofrece ciertas ventajas: evita la creación de ficheros XML para cada una de las peticiones y reduce la carga en la red.

GSL-Charts es, por tanto, un nuevo componente que aprovecha la funcionalidad de generación de gráficas utilizada en *WebChartService*.

Personalización de los datos de las gráficas

Otra de las necesidades era la creación de gráficas personalizando los datos, es decir, de una misma plantilla poder obtener diferentes gráficas finales con diferentes datos.

WebChartService permite la incorporación de los datos en las gráficas añadiéndolos en el propio fichero XML o bien, incluyendo un enlace a otro fichero XML que tenga los datos. Se podría haber elegido cualquiera de estos mecanismos para la creación de las gráficas por parte de la librería pero ya que al disponer de una librería se tiene acceso a la estructura interna de la misma, se decidió la inclusión de los datos directamente sin ficheros intermedios. Para ello se tuvo que estudiar el manejo de datos interno que realizaba *WebChartService* y modificarlo para permitir personalizar los datos sin necesidad de crear un fichero cada vez.

2.5. Polynomial Solver

PolynomialSolver es un servicio web WPS (ver sección F.3) que permite resolver polinomios. Para la resolución de las operaciones matemáticas se apoya en la

aplicación matemática Scilab[3]. Dichas operaciones matemáticas se configurar en un fichero XML que sigue unos esquemas definidos.

Los diseños en detalle de *PolynomialSolver* pueden verse en la sección 2.5.

2.5.1. Trabajo realizado

La remodelación de *PolynomialSolver* ha pasado por optimizar la forma de interactuar que éste tenía con Scilab.

La comunicación con Scilab se realizaba a través de una llamada por consola a un fichero previamente compilado en java, éste método no era adecuado para la depuración, ya que el depurador pierde el control una vez que se llama a la consola, otro inconveniente es que los resultados obtenidos en consola no se pueden gestionar con Java de una manera elegante.

Por tanto se ha investigado la manera en que Scilab puede comunicarse con un proyecto Java para evitar la comunicación a través de consola. Se ha encontrado una solución para utilizar Scilab como API como si de una dependencia cualquiera se tratase, de esta manera no se pierde el control en el depurador y los resultados obtenidos pueden gestionarse sin problemas.

También ha habido que configurar los ficheros XML que definen las operaciones matemáticas necesarias para el cálculo de la infiltración.

2.6. Decision Table Evaluator

DecisionTableEval es un servicio web WPS (ver sección F.3) capaz de resolver tablas de decisión. Una tabla de decisión es una estructura que devuelve un valor en función de si se cumplen un conjunto de condiciones necesarias. Este componente ha sido utilizado para la resolución del cálculo de la infiltración ya que es necesaria la resolución de varias tablas de decisión, sin embargo, no ha sido objeto de modificaciones.

2.7. WebFTManager

2.7.1. Introducción

WebFTManager es una aplicación web genérica para la gestión de información, cuya interfaz está construida apoyándose en el *framework* GWT.

La información en *WebFTManager* se organiza en entidades (*features*), que corresponden con los grupos principales del menú de la izquierda (ver figura 2.5), para cada una de ellas, se pueden declarar grupos de descriptores (*descriptorGroups*), que son las pestañas que aparecen en la parte central de la figura 2.6 y por último para

[3]http://www.scilab.org

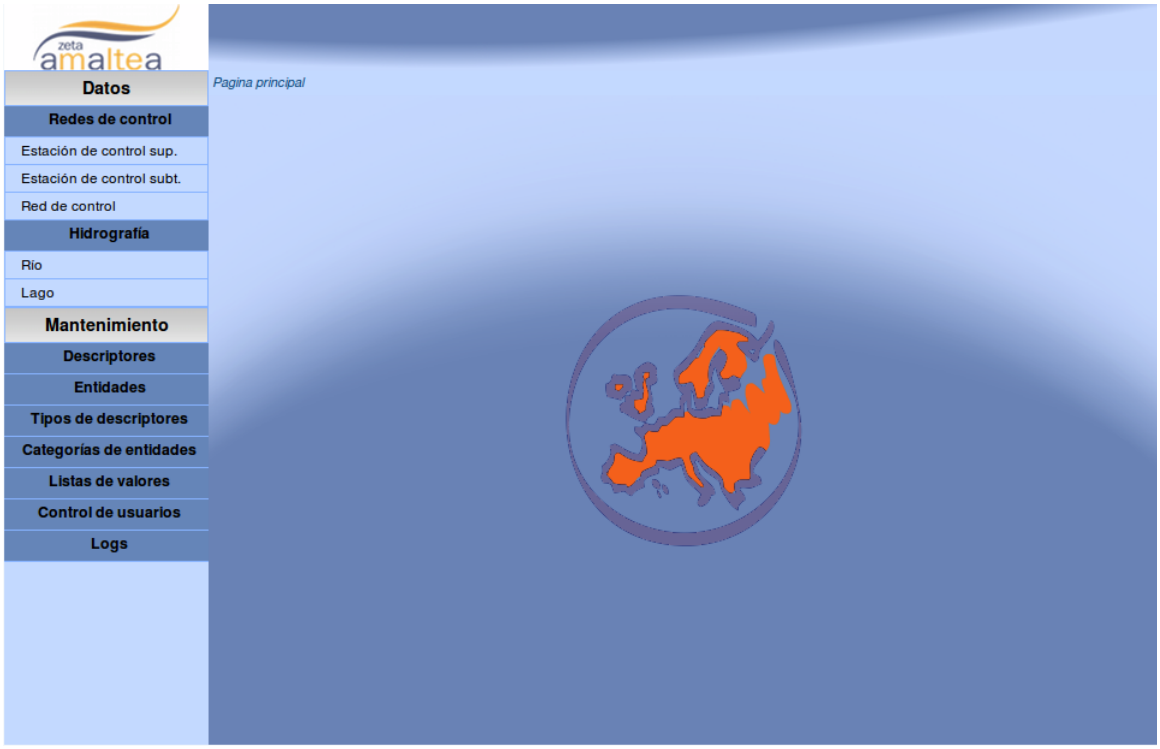

Figura 2.5: WebFTManager

cada uno de esos grupos de descriptores se pueden definir descriptores a mostrar en forma de tabla que pueden apreciarse en la misma figura y que corresponden con cada una de las columnas de la tabla.

Los grupos de descriptores a su vez pueden pertenecer a una de estas clases en función de si sus datos son dependientes o no del tiempo y/o de la profundidad.

- XY: datos que no dependen del tiempo ni de la profundidad
- XYT: series temporales, es decir, datos que dependen del tiempo
- XYZ: series verticales, es decir, datos que dependen de la profundidad
- XYZT: series tempo-verticales, es decir, datos que dependen tanto del tiempo como de la profundidad

Su principal característica es que es capaz de generar la estructura de datos comentada en el párrafo anterior, así como el tipo de dato de cada descriptor, desde la propia interfaz web, generando las tablas necesarias en la base de datos y sus relaciones. De esta manera cada instancia de *WebFTManager* puede ser personalizada con la estructura de datos que se crea conveniente.

Esta aplicación web necesitaba varios cambios para adaptarla a las necesidades de *InfiltrationSolver* que se detallan en la siguiente sección.

Los diseños en detalle pueden verse en la sección C.3.

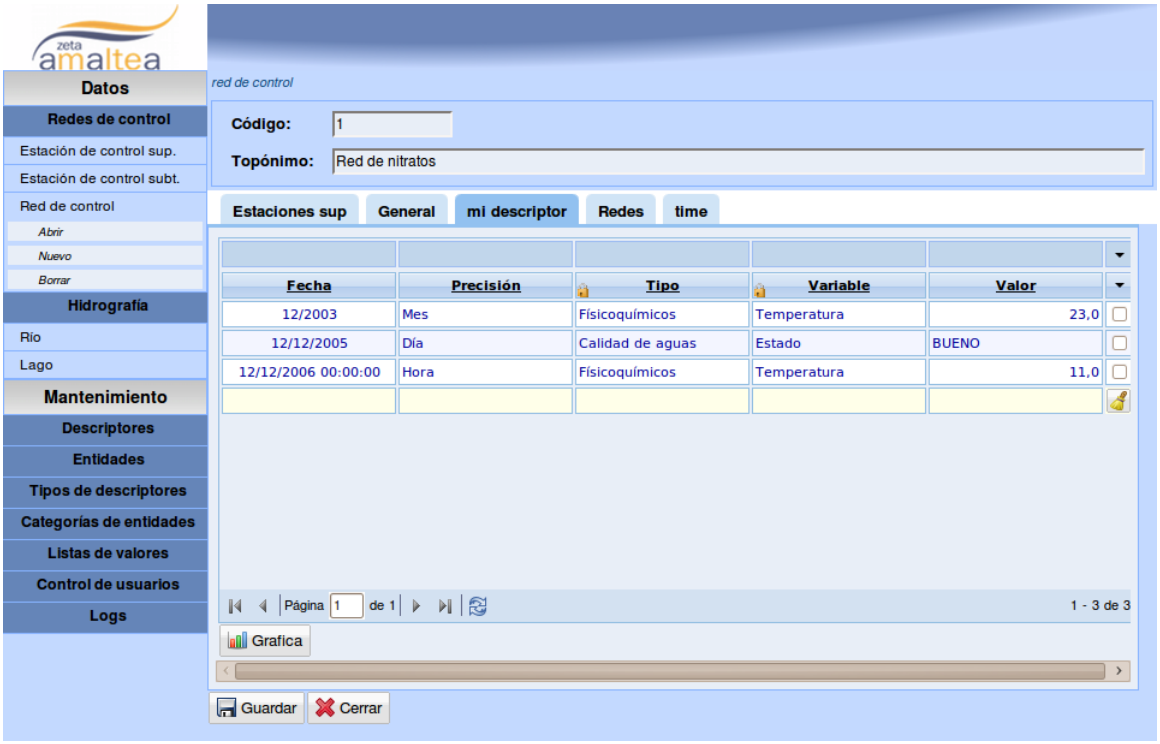

Figura 2.6: Detalle de los grupos de descriptores y descriptores

2.7.2. Trabajo realizado

A continuación se muestran las diferentes tareas realizadas.

Paso de aplicación web autónoma a librería

Una vez que *WebFTManager* había sido elegida como aplicación central para la gestión de los datos, había que pensar como implementar los mecanismos para calcular la infiltración que era el objetivo del proyecto. Como se quería personalizar *WebFTManager* se decidió crear un nuevo proyecto denominado *InfiltrationSolver* (ver sección 2.8), e incluir *WebFTManager* como una dependencia de éste.

Para lograr incluir *WebFTManager* en *InfiltrationSolver*, se tuvo que investigar la manera en que Maven (ver sección F.7) construye los proyectos ya que es con esta herramienta con la que se gestiona la construcción de los mismos.

Teniendo en cuenta que *WebFTManager* podría ser construido como una aplicación web (fichero .war) o una librería (fichero .jar), la primera tarea era permitir al usuario de una manera simple la construcción de uno u otro tipo de fichero. Ésto se consiguió gracias a los perfiles de Maven que permiten personalizar la construcción de los proyectos.

Mejoras en el tratamiento de datos

El cálculo de la infiltración suele realizarse con series largas de datos, por tanto, una de las primeras tareas del proyecto fue modificar *WebFTManager* para que fuese capaz de trabajar con grandes volúmenes de datos sin presentar desbordamientos de memoria y permitiendo al usuario un acceso ágil a la información. Ésto se ha conseguido gracias a dos mecanismos, la tabla paginada y la barra de filtrado.

La **tabla paginada**, en contrapartida a la tabla no paginada, permite cargar la información en porciones denominadas páginas en vez de intentar mostrar todos los datos de vez, de esta manera se evitan desbordamientos de memoria y se mejora el rendimiento de la aplicación. Este componente forma parte de la librería *IAAA_GWT* que aglutina los componentes GWT de los que dispone la empresa.

En la figura 2.7 podemos ver el aspecto de la tabla paginada, la parte central es una tabla donde se muestran los datos y la parte inferior contiene una serie de botones que nos permiten movernos entre las páginas (pasar de página, recargar página...)

Precipitación acum. 5días	Precipitación	Temperatura	Horas sol	ETP	Mes
		7,0375	10,6	1,9235	Febrero
		6,3764	9,1	0,0	Diciembre
		5,8099	9,4	1,9235	Enero
		9,2312	11,9	1,9235	Marzo
		10,9362	13,4	1,9235	Abril
		14,733	14,6	1,9235	Mayo
		21,9391	14,9	0,0	Julio
		9,2034	9,8	1,9235	Noviembre
		22,2505	13,9	1,9235	Agosto
		19,0732	12,9	1,9235	Septiembre
		14,4763	11,1	1,9235	Octubre
		18,7104	15,2	1,9235	Junio

Página 1 de 1 1 - 12 de 12

Figura 2.7: Tabla paginada

Otro de los problemas cuando se trabaja con muchos datos en tablas es tratar de buscar una fila o conjuntos de filas en concreto, de este problema surgió la necesidad de incorporar un mecanismo (barra de filtrado) que permitiese al usuario buscar la información rápidamente.

La **barra de filtrado** es una utilidad que permite al usuario filtrar la información de una tabla según unos criterios. Se decidió implementar con un aspecto similar a una fila de la tabla en la zona superior con un color diferenciador, de tal manera que no resultase molesta al usuario y que a la vez estuviese accesible para él.

En la figura 2.8 podemos ver el el aspecto de una tabla paginada con el prototipo de la barra de filtrado (en la parte superior en color verde).

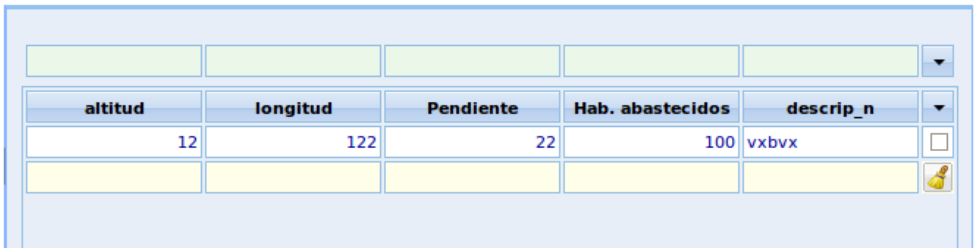

Figura 2.8: Tabla paginada con el prototipo de la barra de filtrado

Las principales características son:

- **Barra de filtrado genérica.**

  Al querer incluir la barra de filtrado en una tabla genérica, debía implementarse de manera que cada celda de la barra de filtrado tuviera un filtro acorde al tipo de dato de la columna correspondiente de la tabla, es decir, si en la columna número uno, se estaban mostrando fechas, el filtro correspondiente a esa columna debía estar asociado a un filtro de tipo fecha, y así con cada uno de los filtros. Este aspecto haría de la barra de filtrado un componente muy flexible.

- **El tipo de filtro depende del tipo de dato.**

  Dependiendo del tipo de dato que se vaya a filtrar, cada filtro tiene opciones diferentes para maximizar la potencia de las búsquedas.

  Un filtro de tipo booleano tiene simplemente como opciones de filtrado (verdadero, falso o todos), mientras que para los tipos de datos numéricos y de fecha era interesante añadir un componente que permitiese no solo filtrar por un número/fecha en concreto, sino ampliar esto, con operadores del tipo {<, <=, >=, > y entre}. De esta manera se pueden realizar búsquedas más generales.

  Para los filtros mencionados anteriormente, además del valor por el que buscar, se decidió, en vez de dejar al usuario la posibilidad de introducir sus filtros en la propia celda en modo texto y, por tanto, hacerle aprender una sintaxis engorrosa e innecesaria, desarrollar dos componentes más que hacen esta labor mucho más agradable y rápida de cara al usuario final y menos propensa a errores de cara a la aplicación. Éstos son, la ventana emergente para el filtro numérico (véase figura 2.9) y la ventana emergente para el filtro de fechas (véase figura 2.10).

  En la figura 2.10 podemos ver una casilla donde introducir una fecha por la que filtrar. Para evitar que la selección de una fecha supusiera un problema de formato (una misma fecha podría escribirse como 17/10/1999 o 17-10-1999) y fuese complicado de manipular y propenso a errores (fechas no existentes), se ha incluido un selector de fecha intuitivo, de tal manera que al hacer clic en la

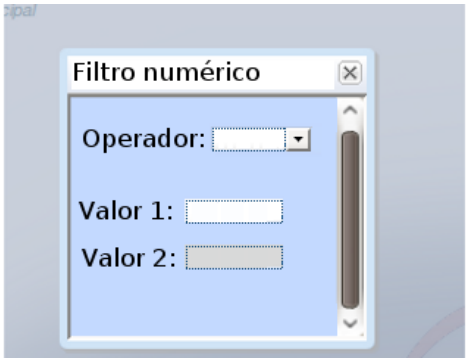

Figura 2.9: Ventana emergente para el filtro numérico

casilla para introducir o modificar una fecha aparece una ventana emergente parecida a un calendario ,véase figura 2.11„ que evita estos inconvenientes.

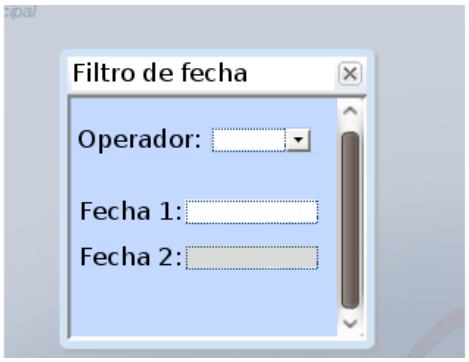

Figura 2.10: Ventana emergente para el filtro de fechas

El resto de los tipos de filtros, estarán asociados al tipo de editor que tenga la celda, si es un menú desplegable, el filtro también será otro menú desplegable con las mismas opciones, permitiendo filtrar por cualquiera de las opciones del menú. Como opción especial, para los filtros de tipo texto, se posibilita la opción de escribir el carácter especial "%". Así si en un filtro de tipo texto introducimos "car%" en la columna número uno, la aplicación devolverá como resultado todas las filas que en la columna número uno contengan una palabra que empiece por "car", (como cara, carpa, etc...)

- **Operaciones en un menú desplegable**

  Un botón a la derecha de la barra de filtrado, véase figura 2.12, despliega un menú donde se aglutinan las opciones más comunes: aplicar filtros (realiza la búsqueda en la base de datos según los valores de filtros elegidos) y limpiar filtros (limpia la barra de filtrado dejándola sin ningún valor en sus celdas).

  Así se mantiene la limpieza en la tabla y se maximiza la flexibilidad de la barra de filtrado para futuras ampliaciones, ya que en caso de querer incorporar alguna nueva opción, añadirla en el menú desplegable sería muy fácil y rápido.

- **Búsquedas con "Y" lógico.**

Figura 2.11: Detalle de la ventana emergente de selección de fechas

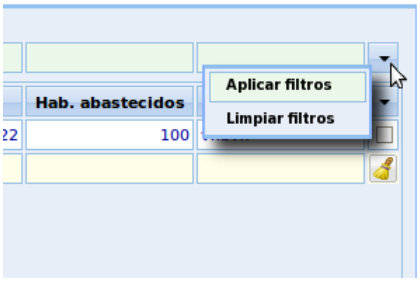

Figura 2.12: Detalle del menú desplegable de filtrado.

Las búsquedas son de tipo "Y" lógico (AND), de tal forma que si seleccionamos un filtro de tipo numérico en la columna número uno con la opción "$<$ 5" y otro de tipo fecha en la columna número 4 con la opción "entre el 15/10/2009 y el 20/10/2009", y pulsamos el botón "aplicar filtros", los resultados mostrados en la tabla después de recibir la respuesta del servidor serán: Todas las filas que en su columna número uno tengan un valor menor que cinco y que además en su columna número cuatro tengan una fecha comprendida entre el quince y el veinte de Octubre de dos mil nueve.

La razón principal de optar por búsquedas con "Y" lógico es que éstas son las más intuitivas y frecuentes aunque se podría incorporar la posibilidad de aplicar filtros con "O" lógico (OR) o aplicando funciones más complejas, selección de conjuntos, múltiples filtros encadenados etc.

Generación de informes

Para la obtención de los informes de las distintas entidades de una manera fácil y rápida se hecho uso de la librería de generación de informes *GSL-Reports* desarrollada durante este proyecto (ver capítulo 2.3).

Un botón en la interfaz nos permite una acceso fácil. Este botón está situado en

cada uno de los formularios que hacen referencia a un tipo de entidad a la que se ha definido una plantilla de BIRT (ver figura 2.13), al pulsar sobre él, se permite elegir un rango de fechas para el que se generará el informe (ver figura 2.14). Una vez que se confirma el rango de fechas el servidor genera el informe que es devuelto al cliente. El cliente solo tendrá que elegir el destino del informe y guardarlo en su equipo.

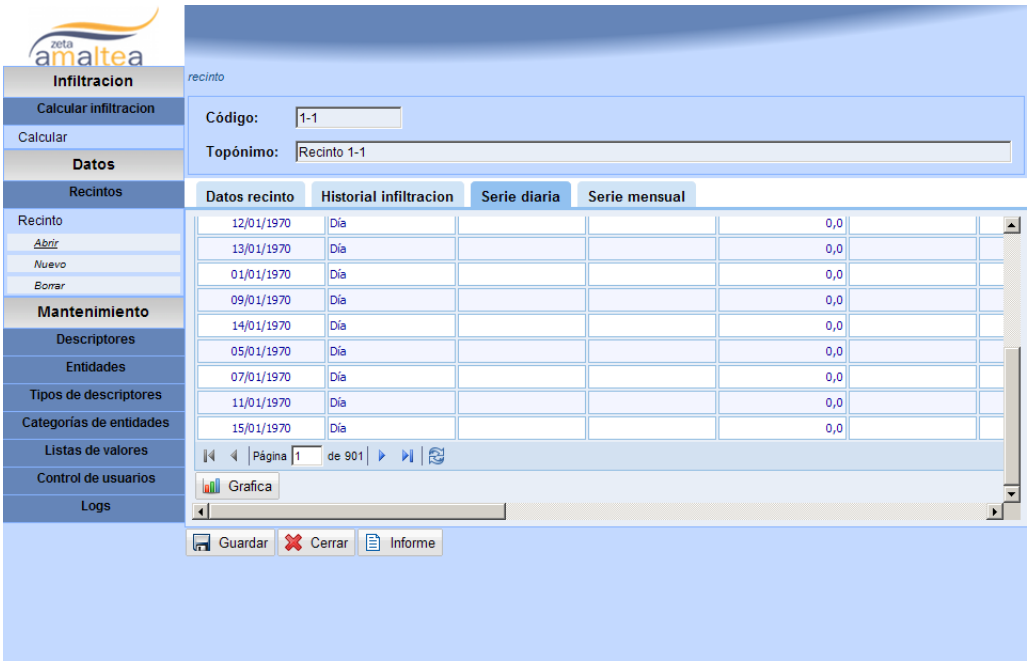

Figura 2.13: Detalle de los iconos de informes y gráficas

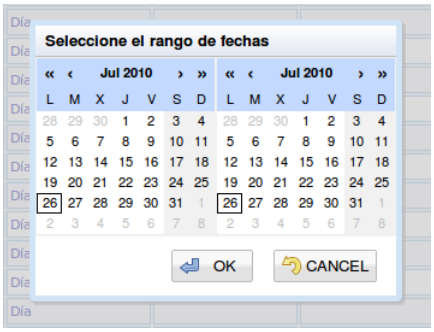

Figura 2.14: Detalle de selección de fechas para un informe

Por último en la figura 2.2 podemos ver un ejemplo de informe generado.

Generación de gráficas

Al igual que en el caso de los informes, se ha hecho uso en *WebFTManager* de la librería de generación de gráficas *GSL-Charts* (ver capítulo 2.4). En *WebFTManager*

RECINTO

Código	4	Topónimo	Recinto 11-1
Número de curva	47	Latitud	42

SERIE DIARIA

Fecha	NC_Corregido	Escorrentia	Umbral_precip	Inf_max	Infiltracion 1	Infiltratición 2	Cond_humedad suelo	Precip_acum	Precipitacion
01/12/2006				0			I	0	0
02/12/2006		0		0			I	6,688	6,688
03/12/2006				0			I	6,688	0
04/12/2006				0			I	0	0
05/12/2006				0			I	0	21,995
06/12/2006				0			I	0	0
07/12/2006				0			II	13,672	13,672
08/12/2006				0			II	16,645	2,972
09/12/2006				117,512			III	39,382	22,738
10/12/2006				117,512			III	45,327	5,944
11/12/2006				117,512			III	45,327	0
12/12/2006				117,512			III	34,627	2,972
13/12/2006				117,512			III	31,655	0
14/12/2006				0			I	8,917	0
15/12/2006				0			I	2,972	0
16/12/2006				0			I	0	0
17/12/2006				0			I	3,715	3,715

26/07/2010 12:20 1 / 2

Figura 2.15: Ejemplo de informe generado

se ofrece un botón que permite, aprovechándose de esta librería, generar gráficas *"al vuelo"*.

Se puede obtener una gráfica de una tabla que tenga una componente temporal, en la figura 2.13 podemos ver un ejemplo de este tipo de tabla. Debajo de los datos aparece un botón con un icono de una gráfica sobre el que se puede pulsar. Al pulsar se lanzará una ventana emergente donde se puede elegir entre las gráficas disponibles para ese tipo de tabla (las gráficas disponibles han tenido que ser previamente diseñadas por el usuario ver anexo D.3.4). En la figura 2.16 se puede ver un ejemplo de selección tanto de gráfica como de rango temporal.

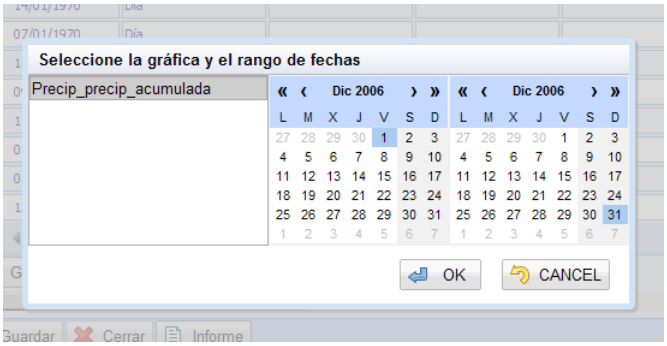

Figura 2.16: Detalle de selección de los parámetros para las gráficas

Al igual que en el caso de los informes, una vez generada esta gráfica con los datos correspondientes, ésta es enviada al cliente como un fichero JPG para ser guardado

en el equipo, tal y como muestra la figura 2.17 .

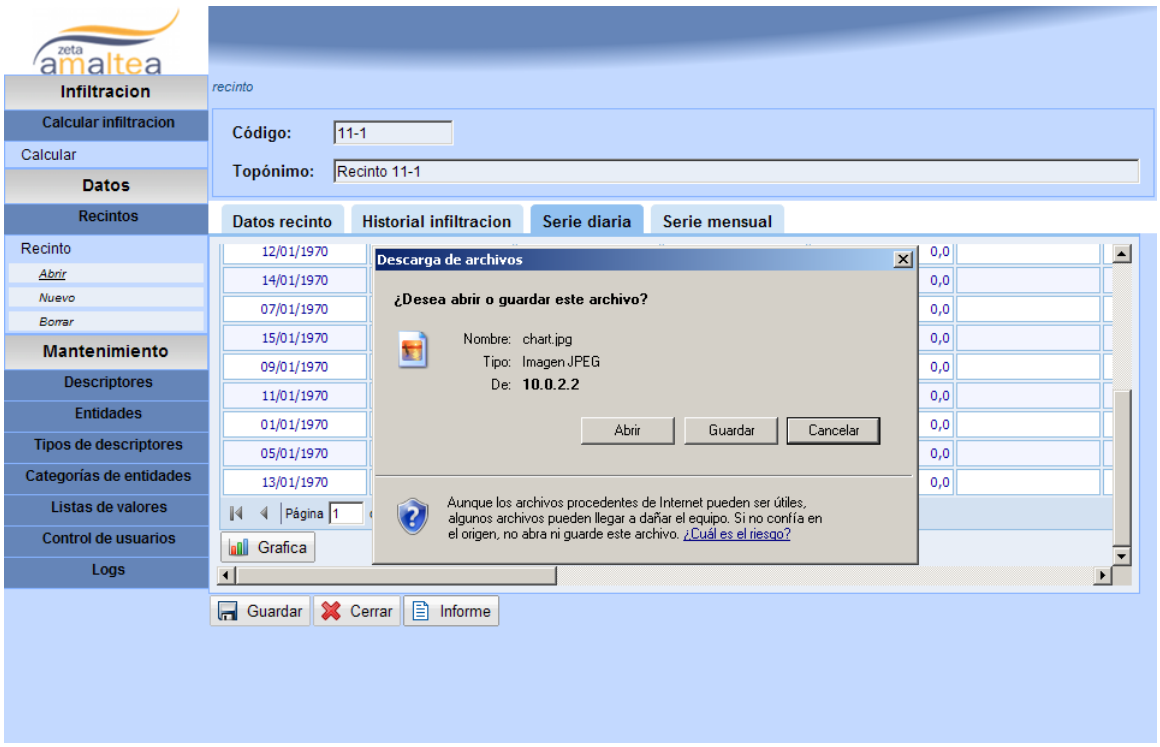

Figura 2.17: Guardado de gráfica

En la figura 2.18 podemos ver un ejemplo de gráfica generada.

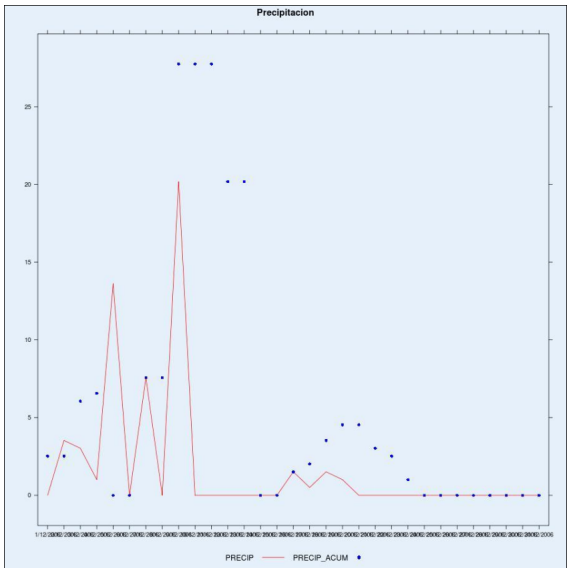

Figura 2.18: Gráfica generada

2.8. InfiltrationSolver

InfiltrationSolver es la aplicación web para el cálculo de la infiltración acuífera, objetivo principal del proyecto fin de carrera. La implementación de esta aplicación se ha basado en la interconexión de los componentes descritos anteriormente de acuerdo a la arquitectura que puede verse en la sección 2.2.

A continuación se resumen todos los componentes y su implicación en *InfiltrationSolver*.

***WebFTManager*:** Es el núcleo principal de la aplicación, tanto la interfaz como los mecanismos de gestión de datos están basados en *WebFTManager* casi en su totalidad, exceptuando los específicos del cálculo de la infiltración que sólo se han implementado en *InfiltrationSolver*

Gsl-Reports: Librería utilizada para la generación de informes "*al vuelo*" a partir de una plantilla prediseñada.

***Gsl-Charts*:** Librería utilizada para la generación de gráficas "*al vuelo*" a partir de una plantilla prediseñada.

***PolynomialSolver*:** Servicio web capaz de resolver las operaciones polinómicas necesarias para el cálculo de la infiltración acuífera. (ver anexo E)

***DecisionTableEval*:** Servicio web capaz de resolver las tablas de decisión necesarias para el cálculo de la infiltración acuífera. (ver anexo E)

2.8.1. Trabajo realizado

La mayoría de la parte funcional ha sido aprovechada de otros componentes genéricos comentados en la sección anterior, no obstante ha habido que realizar varias tareas. Los diseños en detalle de *InfiltrationSolver* pueden verse en la sección C.4.

Configuración

Para adecuar los componentes desarrollados al cálculo de la infiltración ha habido que realizar las siguientes tareas de configuración:

- Configuración de *InfiltrationSolver*: se ha tenido que crear la estructura de datos necesaria para el manejo de los datos de la infiltración: entidades, grupos de descriptores y descriptores. Este proceso se ha realizado a través de la propia interfaz de *InfiltrationSolver* ya que extiende de *WebFTManager*.
- Creación de una plantilla de informes de BIRT: se ha tenido que crear una plantilla de BIRT que muestra "al vuelo" los datos de cualquier recinto.
- Creación de una plantilla de gráficas: se ha tenido que crear una plantilla con el formato de la gráfica a mostrar en *InfiltrationSolver*.

- Definición de los cálculos: tanto la definición de las tablas de decisión de *DecisionTableEval* como la definición de las operaciones matemáticas de *PolynomialSolver* han tenido que ser creados para ajustarse a las necesidades del cálculo de la infiltración acuífera.

Cálculo de la infiltración

Para conseguir que el cálculo de la infiltración acuífera funcionase correctamente, se ha tenido que hacer interoperar todos los componentes que forman *InfiltrationSolver*. Las operaciones necesarias para calcular la infiltración acuífera se definen en el anexo E.3.

En *InfiltrationSolver* una vez que el cliente selecciona el recinto o recintos sobre los que quiere realizar los cálculos, como puede apreciarse en la figura 2.19 , es el servidor el encargado de interoperar con todos los componentes implicados para la realización de dichos cálculos.

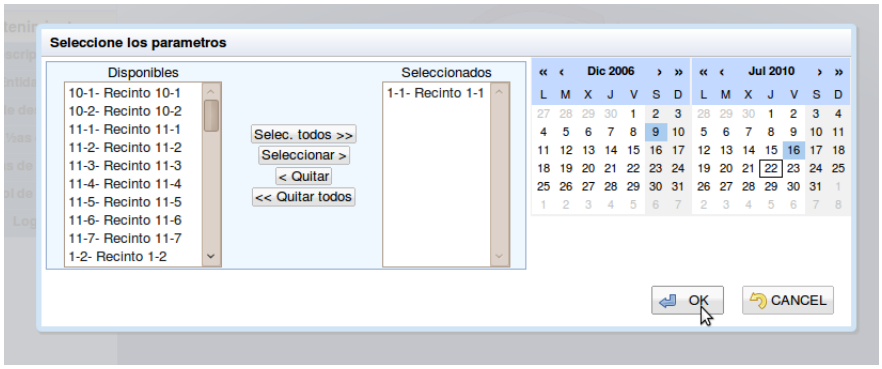

Figura 2.19: Selección de parámetros para el cálculo de la infiltración

Se ha tenido en cuenta que esta operación puede resultar costosa así que en la ventana de progreso (ver figura 2.20) se ha añadido un botón denominado "*background*" que permite esconder esta ventana y permitir al usuario trabajar normalmente. Cuando dichos cálculos finalicen, se enviará un e-mail al usuario que comenzó los cálculos.

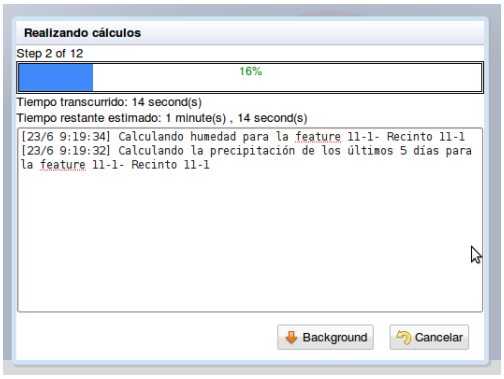

Figura 2.20: Progreso de los cálculos

Capítulo 3

Conclusiones

3.1. Trabajo realizado

Como resultado del proyecto se ha obtenido una aplicación web capaz de capturar los datos de entrada, realizar el cálculo de la infiltración y presentar los resultados por medio de informes, gráficas o en la propia interfaz web. Dicha aplicación ha sido construida en base a componentes reutilizables, algunos ya existían y han sido modificados o mejorados, otros han sido creados desde cero. Además se han creado una serie de componentes genéricos, como la librería de informes, la librería de gráficas y se han mejorado otros que podrán ser reutilizados en otros trabajos. También se ha mejorado el acceso a la información gracias a la barra de filtrado y la tabla paginada.

A modo de resumen podríamos destacar como trabajo realizado:

1. Una librería genérica de generación de informes que permite:
    a) Generación de informes en base a unas plantillas prediseñadas en múltiples formatos diferentes:
    b) Incorporación de imágenes.
    c) Incorporación de mapas de uno o varios WMS con SLD si se desea.
    d) Paso de parámetros para personalizar los informes.
    e) Alterar la fuente de los datos para trabajar con diferentes bases de datos.
2. Una librería genérica de generación de gráficas que permite:
    a) Generar gráficas en base a una plantilla prediseñada.
    b) Incorporar datos al vuelo a una plantilla.
3. Una aplicación web para el cálculo de la infiltración que ofrece:
    a) Interfaz para calcular la infiltración acuífera y visualización del progreso de los cálculos.

4. Mejora y configuración de otros componentes genéricos para el caso concreto de este proyecto

3.2. Lineas futuras

Debido a los requisitos temporales del proyecto ha habido optimizaciones o ideas que se han priorizado en detrimento de otras. Como posibles líneas futuras podríamos destacar:

- **Mejora de prestaciones en el cálculo de la infiltración:** analizar distintos mecanismos para aumentar la velocidad del cálculo de la infiltración como por ejemplo, la ejecución en paralelo de varios cálculos de la infiltración.
- **Personalizar gráficas "al vuelo":** la librería *GSL-Charts* permite generar gráficas de formato predefinido *"al vuelo"*, una posible mejora consistiría en permitir, además, que se pudieran modificar "al vuelo" el diseño (lineas, puntos, sectores...) de las mismas.
- **Tecnología *push* en el servidor:** mientras se realizan los cálculos de la infiltración, el cliente visualiza el progreso de los mismos en pantalla. Este progreso se obtiene preguntando al servidor su estado cada segundo, esta metodología supone una carga en el servidor que puede mitigarse con la implementación de la tecnología *push. Ésta* se basa en el concepto contrario, el servidor es quien informaría al cliente cada vez que hubiese cambios en el progreso de los cálculos.
- **Importación de datos de una hoja de cálculo:** Una posible mejora de cara a evitar la adicción de gran cantidad de datos manera mecánica a través de la interfaz web, sería permitir la importación de los mismos de una hoja de cálculo.
- **Visor de mapas:** Ya que estamos calculando la infiltración de un recinto, otra mejora interesante sería incluir la posibilidad de visualizar la ubicación de dicho recinto en un mapa.

3.3. Conclusiones profesionales y personales

Este proyecto ha sido la culminación y finalización de la carrera de ingeniería en informática. Me ha dado la posibilidad de afianzar los conceptos obtenidos durante la misma así como aprender otros nuevos. He podido profundizar en la ingeniería del software y la metodología de proyectos, lo cual creo que va a ser de gran ayuda en el futuro, así como en programación de lenguaje Java del cual, aunque tenía nociones, no había experimentado un uso tan profundo. Además he aprendiendo el uso de servicios web y la tecnología GWT (Google Web Toolkit).

También se ha tenido contacto con lenguajes de programación muy específicos como R, lenguaje de programación para la aplicación estadística R-Project, o Scilab, lenguaje de programación para la aplicación matemática Scilab. Otras tecnologías utilizadas han sido: BIRT como entorno para la creación de informes y GIT para el control y gestión de versiones. Todo esto me ha permitido tener un abanico muy diverso de tecnologías aprendidas.

Es importante reseñar que este proyecto ha sido realizado en un entorno profesional donde el trabajo en equipo entre los integrantes de la empresa ha sido fundamental para llevar a buen puerto el presente proyecto. Además durante el desarrollo de la librería de informes tuve requisitos temporales ya que se decidió integrar dicha librería en otra aplicación que está a punto de comercializarse. Lo cual me sirvió para afrontar plazos reales de entrega así como la depuración de los errores encontrados.

La realización de este proyecto me ha servido para dar importancia a aspectos que hasta ahora no había tenido tan en cuenta debido a la inferior magnitud de los proyectos a los que me había enfrentado a lo largo de la carrera, como la organización (imprescindible) tanto de proyectos como del software, la necesidad de una metodología de trabajo, la realización de pruebas o la importancia del diseño frente a la implementación.

Capítulo 4

Lista de acrónimos

API Application Programming Interface

BBOX Bounding Box

BIRT Business Inteligence and Reporting Tools

CSS Cascading Style Sheets

GWT Google Web Toolkit

JPG Joint Photographic Group

PDF PostScript Document File

SLD Styled Layered Descriptor

SRS Spatial Reference System

WMS Web Map Server

WMS Web Map Service

WYSIWYG What you see is what you get

WYSIWYM What you see is what you mean

XML eXtensible Markup Language

Anexos

Anexo A

Planificación y control de esfuerzos

A.1. Planificación

A.1.1. Estimación inicial

Inicialmente se realizó una planificación del proyecto, dividiendo el mismo en diferentes tareas y dando una estimación de tiempos a cada una de ellas teniendo en cuenta que la dedicación iba a ser de cuatro horas diarias exceptuando los meses estivales en donde se podría dedicar un máximo de ocho horas diarias. Para mostrar esta planificación de forma gráfica se usa un diagrama de Gantt el cual puede verse en la figura A.1.

Esta planificación inicial se dividía en seis fases que explico a continuación:

- Fase inicial de formación y análisis de requisitos globales.
- Implantación de la barra de filtrado al *WebFTManager*
- Análisis diseño e implementación de la librería de generación de informes.
- Análisis diseño e implementación de la librería de generación de gráficas.
- La fase más compleja y costosa del PFC donde había que hacer interoperar todos los componentes del proyecto.
- Por último, la memoria de este PFC.

A.1.2. Tiempos finales

Los tiempos finales han variado notablemente respecto a la planificada estimación inicial, la razón principal es el primer contacto con la aplicación *WebFTManager*, que resultó ser más duro de lo que se planificó en un primer momento ya que ni la magnitud del proyecto al que se hacía frente (más de 60 clases Java) ni la tecnología utilizada (GWT) tenían comparación con ningún otro abordado durante la carrera.

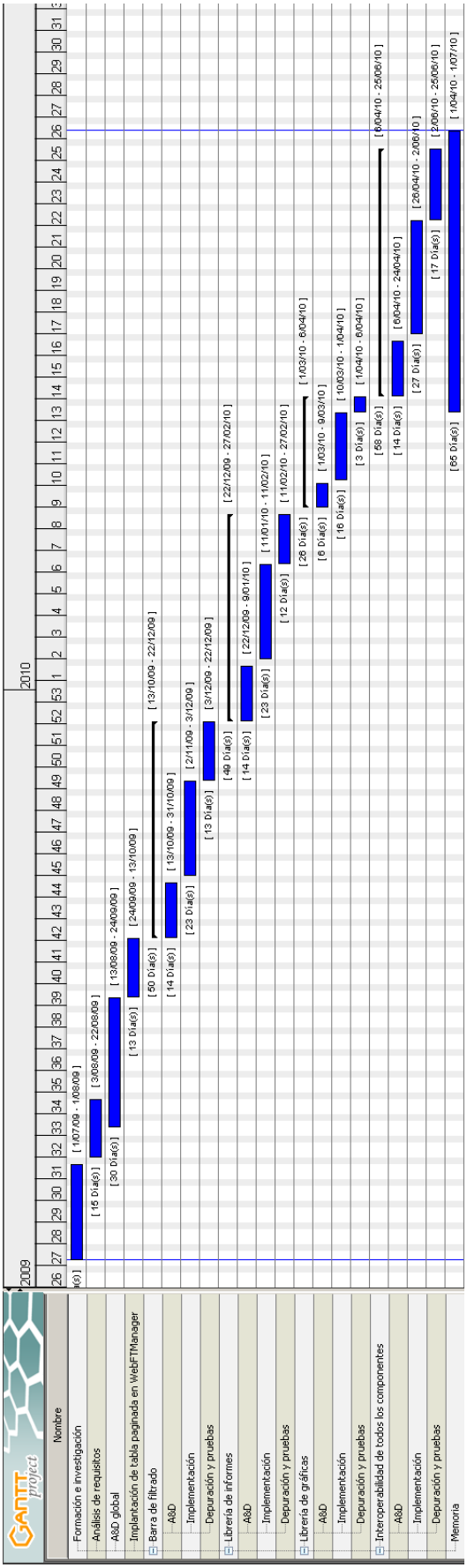

Figura A.1: Diagrama de Gantt inicial

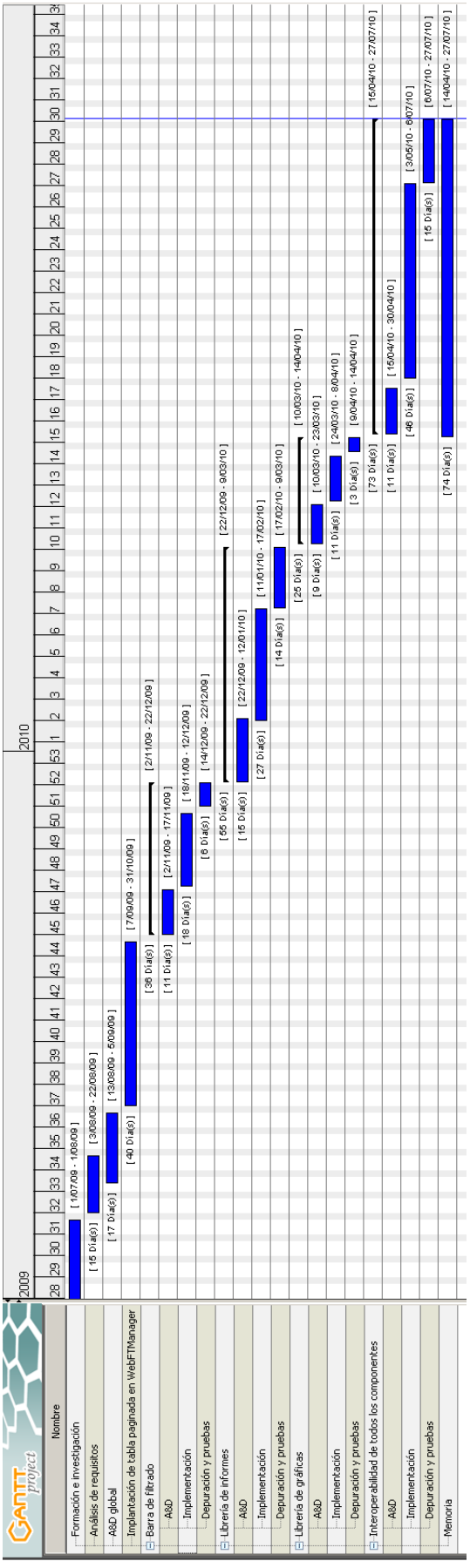

Figura A.2: Diagrama de Gantt final

Además, conseguir la interoperabilidad de los componentes ha sido más compleja y costosa de lo previsto, ya que se ha tenido que modificar y mejorar alguno de ellos para su correcta reutilización. Todo ello sumado a pequeños desfases admisibles ha retrasado la finalización del proyecto.

El resto de las tareas han seguido una línea parecida a la planificada, con variaciones poco significativas. En la figura A.2 podemos ver el diagrama de Gantt resultante a la finalización del PFC.

A.2. Control de esfuerzos

Durante todo el proyecto se han ido apuntando en una hoja de cálculo las horas que se iban dedicando a cada uno de los proyectos y más concretamente a qué parte del mismo (análisis y diseño, implementación...), así como una descripción de qué se había hecho con más detalle ese día.

Las tareas realizadas se han agrupado posteriormente en categorías que se describen a continuación:

- Investigar: Horas invertidas en formación e investigación de nuevas tecnologías o metodología de trabajo.
- Reunión: Horas dedicadas a reuniones con la directora de proyecto o compañeros de GeoSpatiumLab.
- Análisis y diseño: Horas invertidas en el proceso de análisis y diseño de los componentes del proyecto.
- Implementación: Horas dedicadas a la programación de los componentes.
- Documentación: Horas invertidas en la redacción de documentos tanto internos (manual de usuario, manual de desarrollador) como a esta memoria.
- Pruebas: Horas dedicadas a la comprobación del correcto funcionamiento de los componentes así como a su depuración.

En la siguiente tabla A.1 podemos ver las horas dedicadas a cada una de las diferentes tareas, en la figura A.3 podemos ver los mismos resultados de forma gráfica.

Tarea	Horas	Porcentaje
Investigar	149	12,51 %
Reunión	27	2,26 %
Análisis y diseño	42	3,52 %
Implementación	548	46,01 %
Documentación	185	15,53 %
Pruebas	240	20,15 %
Total de horas	**1191**	**100 %**

Tabla A.1: Horas empleadas por tarea

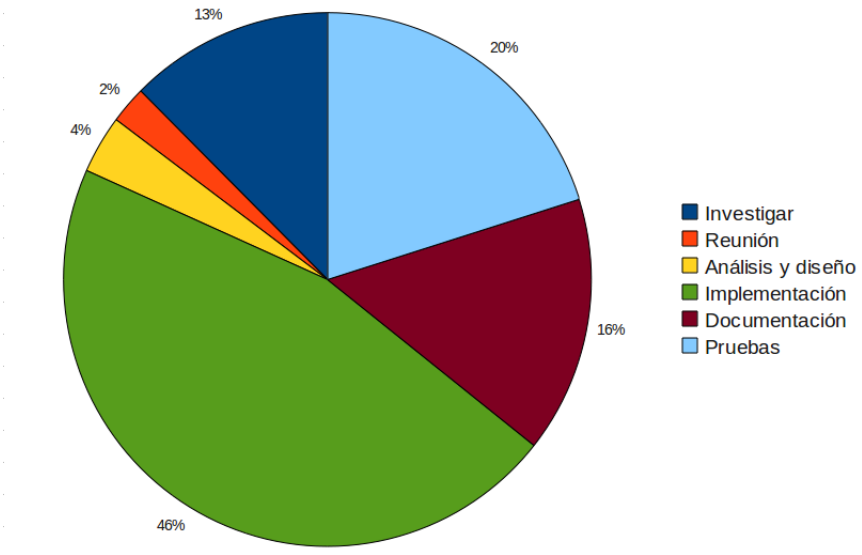

Figura A.3: Gráfica de distribución de horas por tarea

Mes	Horas
Julio	149
Agosto	121
Septiembre	97
Octubre	80
Noviembre	77.5
Diciembre	93
Enero	35
Febrero	94
Marzo	92
Abril	64
Mayo	91
Junio	96
Julio	101.5

Tabla A.2: Horas empleadas por mes

Este proyecto se ha realizado en algo más de 12 meses, desde principios de Julio de 2009 hasta principios de Agosto de 2010. La distribución del esfuerzo realizada se ve reflejada en la tabla A.2 y de forma gráfica en la figura A.4.

Puesto que se ha combinado el trabajo de este proyecto con el desarrollo de asignaturas de la universidad, la dedicación al proyecto en los meses lectivos ha sido a media jornada, mientras que los meses estivales se ha dedicado la jornada completa. En caso de que el proyecto hubiese sido realizado a tiempo completo íntegramente, la duración estimada sería de unos 6-8 meses. Sumando las horas dedicadas en cada mes, se obtienen las horas totales dedicadas a la realización de este proyecto dando un total de 1191 horas.

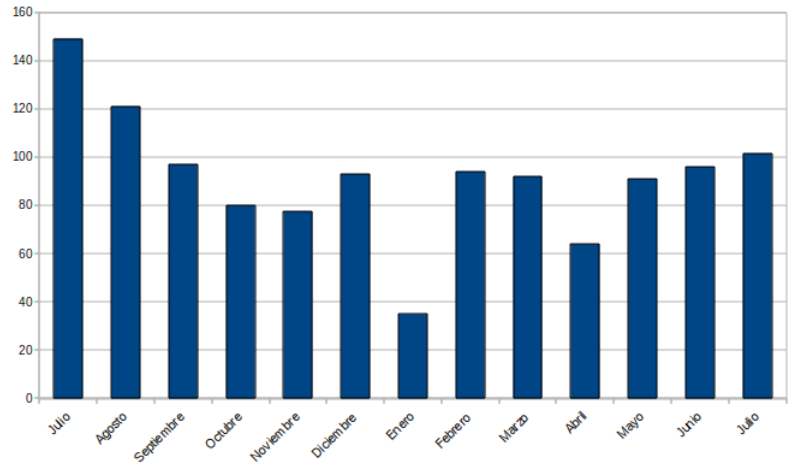

Figura A.4: Gráfica de horas invertidas cada mes

Anexo B

Metodología de trabajo

En este capítulo se abordan las cuestiones referentes a la gestión del proyecto. Lenguaje de programación elegido, entorno de desarrollo y herramientas empleadas, tecnologías utilizadas, metodología de desarrollo seguida y gestión del código.

B.1. Gestión del proyecto

A la hora de afrontar un proyecto software de un tamaño considerable, es necesario el uso de alguna metodología de trabajo que sirva de guía para estructurar y organizar el trabajo a realizar. Normalmente en un proyecto software se suele trabajar en grupo donde la necesidad de organización y gestión es crucial y se acentúa la necesidad de esta labor.

En este proyecto se ha trabajado en solitario, pero no por ello se ha dejado este aspecto de tratar.

Para este proyecto se ha seguido una metodología en cascada, en la que se han seguido las siguientes etapas:

- Investigación y documentación inicial
- Análisis de requerimientos
- Diseño del sistema
- Implementación del código
- Pruebas

Durante las etapas se organizaban reuniones periódicamente para vigilar la correcta evolución de los componentes sobre todo en las fases iniciales, para fomentar ideas de diseño así como para detectar y subsanar posibles errores.

B.2. Gestión de configuraciones

En el transcurso de la etapa de implementación, dada la magnitud del proyecto y que en algunos componentes (*IAAA_GWT, GSL-Reports*) se ha trabajado en común, se requiere de un sistema de control de versiones para facilitar el trabajo y el progreso de la implementación. Se ha usado SubVersion y GIT para esta labor y Maven para la construcción de los proyectos.

B.2.1. Control de versiones

Se ha utilizado SubVersion en un primer momento para posteriormente migrar a GIT para el control de versiones.

Ambos mantienen un registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forma un proyecto y permite que distintos desarrolladores colaboren.

La razón principal para migrar de SubVersion a GIT ha sido que éste último, a diferencia de SubVersion que es un sistema de control de versiones centralizado, es un sistema de control de versiones distribuido, ésto deriva en varias ventajas muy interesantes respecto a SubVersion:

- No depende de acceso a la red para trabajar: todo el repositorio se almacena en el disco duro del cliente, con lo que ante un fallo de red, se puede seguir utilizando el repositorio.

- Gran velocidad de creación de ramas y manejo de proyectos grandes: las operaciones utilizan los recursos locales que son varios órdenes de magnitud más rápidos que la velocidad de conexión.

- Optimización del espacio físico y del tráfico de red: Cuando se crea una rama nueva, GIT trabaja con punteros no duplica los ficheros con lo que no duplica el espacio. Además el tráfico de red se ve también disminuido.

- Las copias de seguridad son algo trivial: Al ser cada cliente un repositorio en sí mismo, aunque exista un repositorio "central" ante un fallo de éste, cualquier cliente mantiene una copia de seguridad.

B.2.2. Maven

La labor de construcción de los proyectos puede llegar a convertirse en un problema para proyectos con un mínimo de tamaño. Maven es una herramienta software que facilita esta labor. En el anexo F.7 se expande y explica esta información.

B.3. Programación

B.3.1. Java

Para el desarrollo del proyecto se ha utilizado el lenguaje de programación Java [?], desarrollado por Sun Microsystems [?] a principios de los años 90. Java es un lenguaje de programación orientada a objetos, con una sintaxis similar a la de C++. Los programas Java se ejecutan sobre una máquina virtual a través del bytecode, un código intermedio más abstracto que el código máquina. En 2007 Sun liberó el código fuente de la plataforma Java, convirtiéndolo en Software Libre casi en su totalidad. La elección de Java frente a otras alternativas se debe a que las aplicaciones y servicios web se han desarrollado en conjunto con GWT (descrito en el anexo F.6). Las principales características que ofrece Java son portabilidad, orientación a objetos, gestión de memoria dinámica transparente al desarrollador, inmenso número de utilidades y tecnologías disponibles y, actualmente, es el principal lenguaje para el desarrollo de software empresarial.

B.3.2. Patrones de diseño

A la hora de abordar un problema de diseño y modelado en el desarrollo de software, suelen producirse muchos problemas con similares características. Mediante el uso de patrones de diseño se pretende dar soluciones a problemas de diseño de unas características similares. Patrones de diseño identificados y utilizados en los diferentes componentes creados son los siguientes:

Façade: mediante el uso de este patrón se establece que un determinado objeto actúe como *fachada* de un conjunto de elementos, es decir, como entrada a la funcionalidad que implementan un conjunto de objetos. Es usado en la librería de generación de gráficas para ofrecer la funcionalidad del componente mediante una única clase que será usada por los clientes del componentes.

Otros patrones de diseño con los que se ha tenido contacto ya que estaban implementados en otros componentes son:

Model View Controller: es un patrón de sistema apto para estructurar o o una aplicación que requiere de una interfaz de usuario. Sus tres siglas designan lo o siguiente:

- Modelo: consiste en el contenido propiamente dicho, esto es, los datos, y los mecanismos de los recuperan y operan con ellos.
- Vista: La manera en que los datos están representados visualmente, y la manera en que se recogen los eventos producidos por el usuario.
- Controlador: La interacción entre el modelo y la vista, y viceversa.

Este patrón está implementado en la librería IAAA_GWT que reúne multitud de componentes GWT muy utilizados en la empresa, concretamente en la tabla paginada.

AbstractFactory: El patrón de creación AbstractFactory es útil en situaciones en las que interesa crear familias de objetos similares, en las que radica una diferencia contextual que implica que internamente se implementan de manera diferente. Un ejemplo típico es la creación de una misma interfaz para diferentes entornos: los objetos a crear deben compartir aspecto y funcionalidad, pero evidentemente cada uno de los entornos tiene sus particularidades. Este patrón ha sido utilizado en *PolynomialSolver.*

B.3.3. Manejo de errores

La gestión de errores es un aspecto primordial en cualquier proyecto. Se han usado los mecanismos de Java para el control de errores para la creación de excepciones propias. Además las partes de código fuente que utilizan GWT, soportan el paso de excepciones entre la parte cliente y servidor, por lo que no se ha tenido que cambiar la forma de tratamiento de excepciones en este ámbito. Los mecanismos utilizados han sido.

Uso de excepciones comunes: Una excepción indica un error que se ha producido en un punto concreto del código, mediante este procedimiento se pueden conocer errores de uso de determinadas clases o componentes, errores inesperados del entorno o errores en las entradas.

Uso de excepciones de tiempo de ejecución: Una excepción de tiempo de ejecución es un tipo de excepción especial que se produce de una forma inesperada y no deseada. En caso de producirse se deberá a un fallo en la programación del componente que la generó. Un caso típico que muestra este comportamiento es la excepción *NullPointerException,* que se produce cuando se intenta usar una variable sin instanciar, es un error que no se esperaba obtener y será debido a un fallo de la programación. Mediante el uso de estas excepciones se permite describir de una forma fácil y rápida nuevos fallos en el sistema, pudiendo incluso realizar un plan para la notificación de éstas.

B.3.4. Registro de ejecución

El registro de ejecución de las aplicaciones supone una de las principales vías para el desarrollador para la depuración de la aplicación aun cuando no es fácil reproducir la situación que produjo el error. En este proyecto se ha hecho uso de una librería de *log* propia de GeoSpatiumLab que almacena la ejecución de la aplicación, tanto si se realiza de forma correcta como si hay algún error, dicha librería genera dos ficheros:

STATS: En este fichero se almacenan los pasos de una ejecución correcta de la aplicación.

Un ejemplo de este fichero:

```
[07-05-2010_10:08:47] Iniciando la aplicación ...
[07-05-2010_10:08:48][DESCRIPTORS_LOAD] Loaded descriptors
[07-05-2010_10:08:49][FEATURETYPES_LOAD] Loaded categories
```

```
[07-05-2010_10:08:49][FEATURETYPES_LOAD] Loaded feature type with id = 72.
[07-05-2010_10:08:53][FEATURETYPES_LOAD] Loaded feature type with id = 44.
[07-05-2010_10:08:53][FEATURETYPES_LOAD] Loaded feature type with id = 46.
[07-05-2010_10:08:53][FEATURETYPES_LOAD] Loaded feature type with id = 10.
[07-05-2010_10:08:53][FEATURETYPES_LOAD] Loaded feature type with id = 22.
[07-05-2010_10:08:53][FEATURETYPES_LOAD] Loaded feature type with id = 68.
[07-05-2010_10:08:54][FEATURETYPES_LOAD] Loaded feature type with id = 48.
[07-05-2010_10:08:54][FEATURETYPES_LOAD] Loaded feature type with id = 82.
[07-05-2010_10:08:54][FEATURETYPES_LOAD] Loaded feature type with id = 84.
[07-05-2010_10:08:54][FEATURETYPES_LOAD] Loaded feature type collection ---
```

DEBUG: En este fichero se almacenan los errores y excepciones que se generaron durante la ejecución.

Un ejemplo de este fichero:

```
[06-07-2010_12:42:49]<<< Open Manager: Error al obtener el parámetro max_session
[06-07-2010_13:02:00]<<< Open Manager: Error al obtener el parámetro max_session
[06-07-2010_13:21:07][FEATURE_LOAD] Error loading mispruebas whith key 1
(SQL query: SELECT PK,DESCRIPTOR_ID,VALUE_INT,VALUE_REAL,VALUE_BOOLEAN,VALUE_STRING,
VALUE_DATE,VALUE_MEMO FROM MISPRU_X_EST_SUP_DATA WHERE FT1 = 1AND
FT2 = 1 ORDER BY DESCRIPTOR_ID ASC).
[06-07-2010_13:21:07]java.sql.SQLException: ORA-00904: "VALUE_BOOLEAN": id no válido
oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:134)
oracle.jdbc.ttc7.TTIoer.processError(TTIoer.java:289)
oracle.jdbc.ttc7.O all7.receive(O all7.java:582)
oracle.jdbc.ttc7.TTC7Protocol.doO all7(TTC7Protocol.java:1986)
oracle.jdbc.ttc7.TTC7Protocol.parseExecuteDescribe(TTC7Protocol.java:880)
oracle.jdbc.driver.OracleStatement.doExecuteQuery(OracleStatement.java:2516)
oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:2850)
```

B.3.5. Entorno de desarrollo

Para el desarrollo Java se ha elegido el entorno de desarrollo Eclipse, un editor con soporte para coloreado de código, autocompletado, refactorización, *debugging*, etc... Además permite la integración de plugins externos que completen su funcionalidad. Para este proyecto se han utilizado:

- Subclipse , como cliente de Subversion integrado en el propio Eclipse.
- JGIT, como cliente de GIT integrado en el propio Eclipse.
- BIRT Designer, para realizar las plantillas de los informes.

B.4. Otras herramientas

Para la realización de esta memoria se ha utilizado LyX [1]. LyX es un procesador de textos que fomenta para la escritura un enfoque basado en la estructura del

[1] http://www.lyx.org/

documento WYSIWYM ("lo que ves es lo que quieres decir") y no simplemente en su aspecto WYSIWYG ("lo que ves es lo que obtienes"). De esta manera la elaboración del documento está centrada en el contenido, sin preocuparse del formato del mismo, como suele pasar en herramientas WYSIWYG donde se requiere de una elaborada habilidad para el formateo de documentos, labor que se complica en magnitudes excepcionales mientras se va aumentando el tamaño del documento.

LyX combina la potencia de TeX/LaTeX con la facilidad de uso de una interfaz gráfica consiguiendo unos resultados visuales profesionales.

Otras herramientas que se han utilizado aunque se consideran secundarias son:

- GIMP: editor gráfico para el diseño de iconos de la aplicación y las imágenes de esta memoria.
- Gantt Project: para la elaboración de gráficos de gantt usados para la planificación a medio y largo plazo.
- Dia: para la creación de diagramas.

Anexo C

Análisis y diseños en detalle

En este anexo se van a describir los análisis y diseños de los diferentes componentes desarrollados en este proyecto.

C.1. GSL-Reports

C.1.1. Requisitos

Requisitos funcionales

Los requisitos funcionales definen las acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados.

- Generación del informe en diferentes tipos de datos
- La librería deberá permitir incorporar imágenes
- La librería deberá permitir incorporar mapas, desde uno o varios WMS con la opción de SLD
- La librería debe proporcionar mecanismos para personalizar los datos de un informe.

C.1.2. Casos de usos

En la figura C.1 se muestran los casos de uso de la librería *GSL-Reports* que se explican a continuación.

Añadir mapa: Cuando se quiere añadir un mapa, primero se consulta a los servidores de mapas correspondientes y se obtiene la imagen final, y después se llama a añadir imagen.

Añadir imagen: Añade una imagen al informe

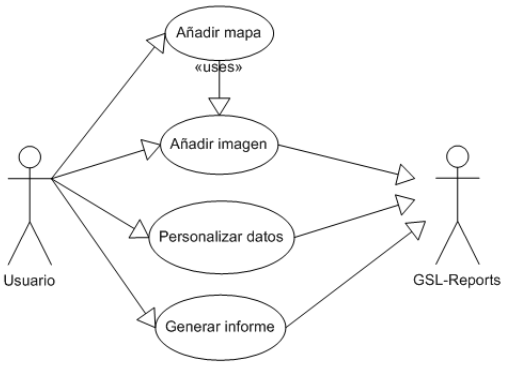

Figura C.1: *GSL-Reports* - Casos de uso

Generar informe: Crea el fichero final en formato PDF, DOC o incluso lo envía como un flujo de bytes.

Personalizar datos: Permite personalizar datos del informe (conjunto de datos, de fuentes, estilo css..)

C.1.3. Diagramas de secuencia

En la figura C.2 podemos apreciar la representación de los casos de uso mencionados en el apartado anterior.

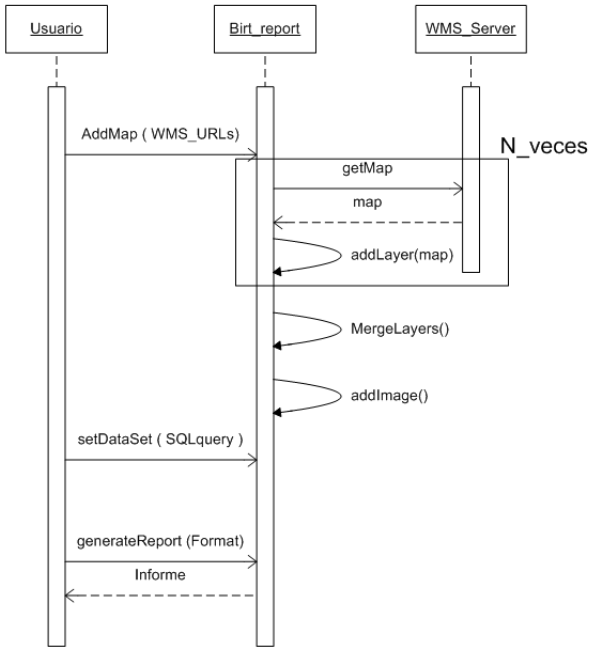

Figura C.2: *GSL-Reports* - Diagrama de secuencia

La primera operación es la inclusión de un mapa que contiene capas de diferentes servidores, por lo que primero realiza todas las peticiones y obtiene las capas, posteriormente las fusiona en una única imagen que es la que se añade al informe.

La segunda, es una operación de personalización ya que modifica el conjunto de datos con los que trabajará el informe. Existen más operaciones de personalización de datos (ver sección D.2).

Y por último se genera el informe en el formato que nos interesa.

C.1.4. Diagrama de clases

En la figura C.3 se puede ver el diagrama de clases creado durante la fase de diseño de *GSL-Reports*, la clase principal es BirtReport que extiende de una más general llamada Report, en la clase Report se implementan operaciones independientes del motor de informes como la obtención de la imagen final a partir de las distintas peticiones de capas a los servidores de mapas y se obliga a la implementación de otras en las clases que extiendan de ella (como generateReport). Mantiene comunicación con Utils (clase de utilidades) y con el paquete de excepciones para el manejo de errores.

BirtReport especializa Report y la adapta según las características específicas del motor de informes (BIRT en este caso).

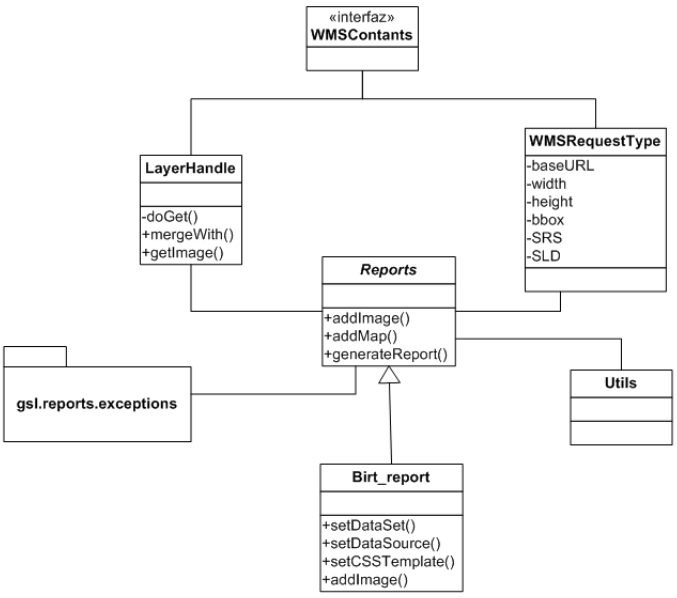

Figura C.3: *GSL-Reports* - Diagrama de clases

C.2. GSL-Charts

C.2.1. Requisitos

Requisitos funcionales

- Generación de gráficas a partir de ficheros XML con datos y configuración
- Generación personalizada de gráficas al añadir los datos en tiempo de ejecución

C.2.2. Casos de usos

En la imagen C.4 pueden verse los casos de uso que se explican a continuación:

Generar gráfica: El usuario genera una gráfica en base a un fichero XML.

Añadir datos: Adicción de datos a una gráfica que haya sido definida sin datos en su fichero XML.

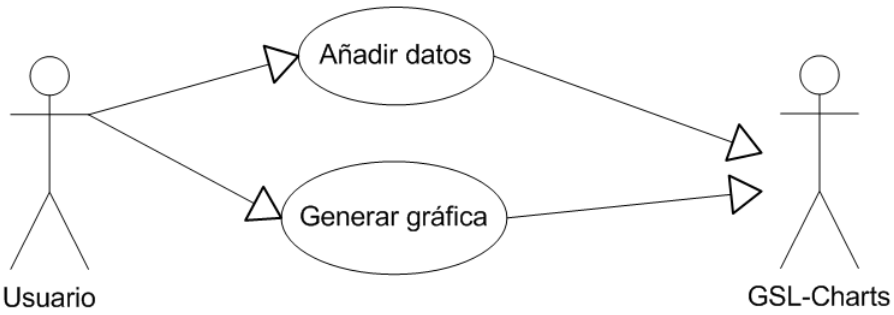

Figura C.4: *GSL-Charts* - Casos de uso

C.2.3. Diagramas de secuencia

Los casos de usos explicados en la sección anterior se pueden ver gráficamente en el diagrama de secuencia de la figura C.5

En la primera operación el usuario introduce la gráfica prediseñada, ésta puede contener en su interior tanto los datos como el formato de la gráfica, en caso de que no tuviera datos, se pueden introducir en tiempo de ejecución con la siguiente llamada "setData", a partir de ahí se puede generar la gráfica.

Cuando se llama a generateChart, la llamada interna buildChart es la encargada de proveer de datos a la llamada final createChart, bien sean desde el propio

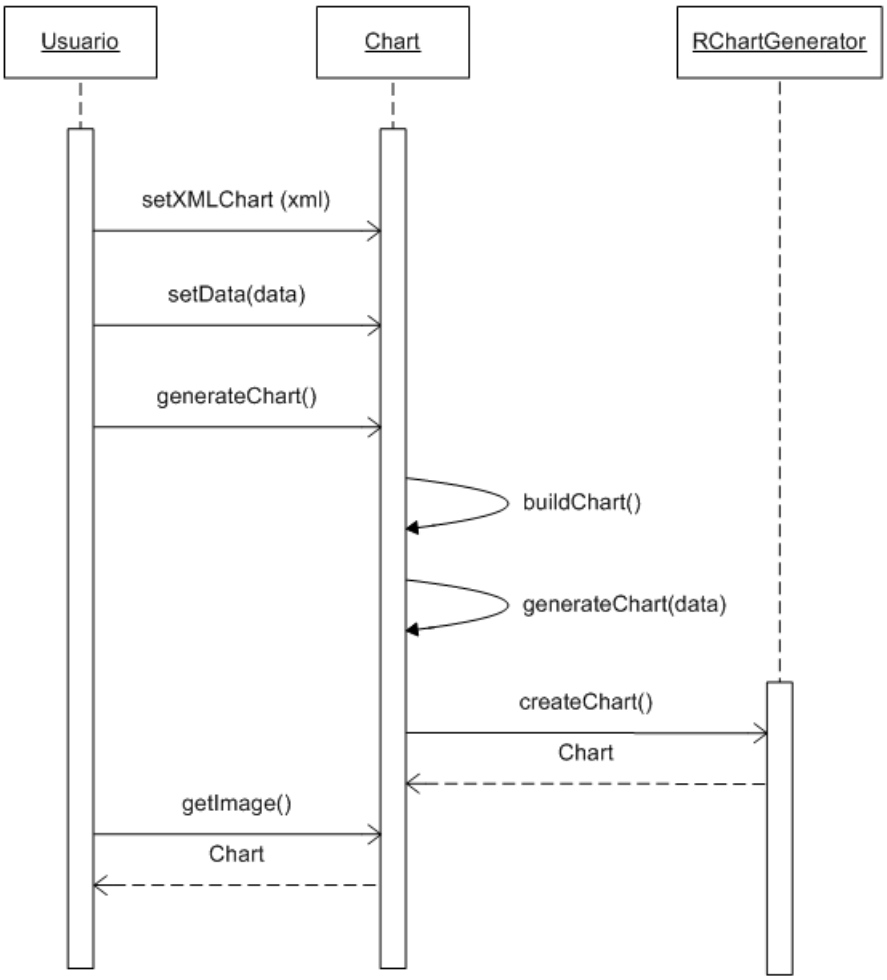

Figura C.5: *GSL-Charts* - Diagrama de secuencia

XML de la gráfica o los introducidos por el usuario, la secuencia termina llamando a RChartGenerator que es la clase que interacciona con R y devuelve la gráfica generada.

C.2.4. Diagrama de clases

El diagrama de clases completo se muestra en la figura C.6 , la clase principal *Chart* es el punto de entrada de la librería y se ha creado para permitir la interacción sencilla con el usuario, el resto de clases han sido aprovechadas del servicio web existente *WebChartClient.*

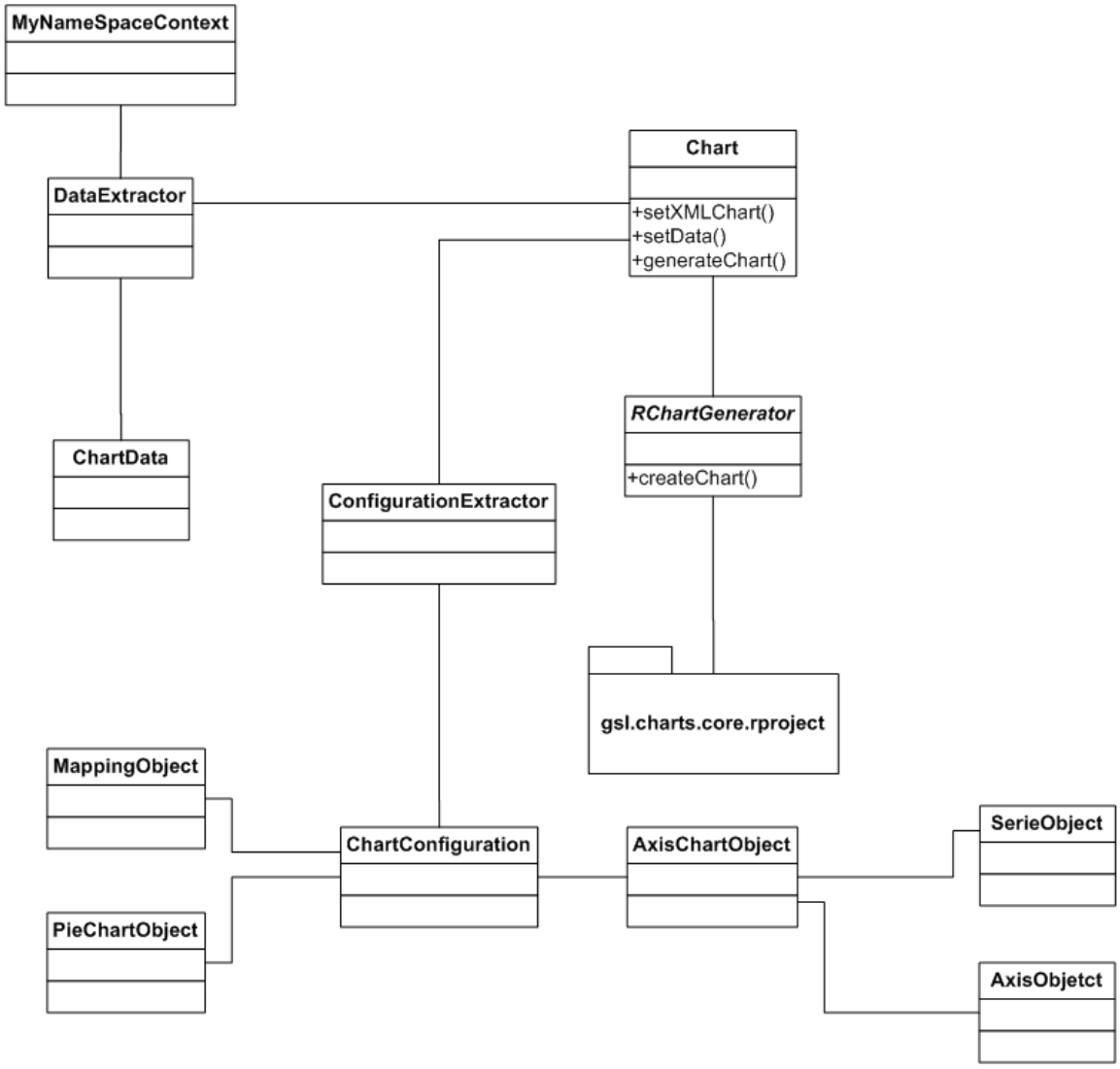

Figura C.6: *GSL-Charts* - Diagrama de clases

C.3. WebFTManager

WebFTManager no ha sido desarrollado como tal, sino que se le han ido agregando funcionalidades nuevas, aquí se detallan los análisis y diseño de algunas mejoras en detalle.

C.3.1. Requisitos

Requisitos funcionales

Los requisitos funcionales definen las acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados.

- La aplicación tratará convenientemente series grandes de datos evitando desbordamientos de memoria
- Se creará un botón para crear informes desde la interfaz, este botón solo estará disponible en las entidades para las que haya sido definida una plantilla de BIRT.
- Los datos de los informes se podrán restringir con un selector de fechas.
- El formato de documento obtenido al generar un informe será PDF.
- Se creará un botón para crear gráficas desde la interfaz, este botón estará disponible en todas las tablas de tipo temporal.
- Al pulsar sobre un botón de generación de gráficas, se dará la opción de elegir una de las gráficas preconfiguradas disponibles así como el rango de fechas de los datos.

Requisitos no funcionales

- En caso de utilizar librerías o aplicaciones externas, la licencia de utilización debe ser libre.

C.3.2. Casos de uso

En la figura C.7 se muestran los casos de uso añadidos tras el trabajo realizado en el componente *WebFTManager*

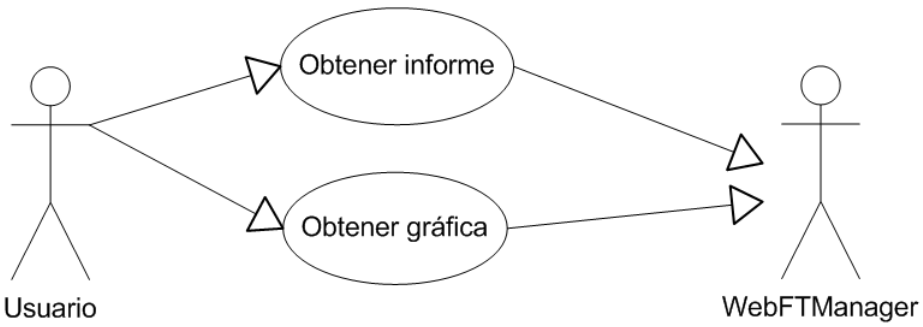

Figura C.7: *WebFTManager* - Casos de uso

Obtener un informe: El usuario gracias a un botón en la interfaz, crea un informe "*al vuelo*" entre unas fechas especificadas, éste es enviado al cliente quien podrá guardarlo en su equipo

Obtener una gráfica: El usuario gracias a un botón en la interfaz, crea una gráfica "*al vuelo*" entre unas fechas especificados, ésta es enviada al cliente quien podrá guardarla en su equipo

Nota: Cabe destacar que los casos de uso de *WebFTManager* son mucho más variados, pero se muestran únicamente los desarrollados a lo largo del proyecto.

C.3.3. Diagramas de secuencia

Dichos casos de uso pueden verse gráficamente en el siguiente diagrama de secuencia.

Obtener un informe: En el caso de la obtención de un informe (ver figura C.8), una vez que se ha pulsado el botón que hay disponible en *WebFTManager* para la generación de un informe, se llama a buildEnclosurePanel, se selecciona el rango de fechas para el informe y se envían al servidor, en este caso al servidor de descargas, que se pone en comunicación con WFTM servlet para la generación del informe, una vez que el servidor obtiene el informe, lo devuelve al servidor de descargas y éste lo devuelve al cliente para que lo almacene en su equipo.

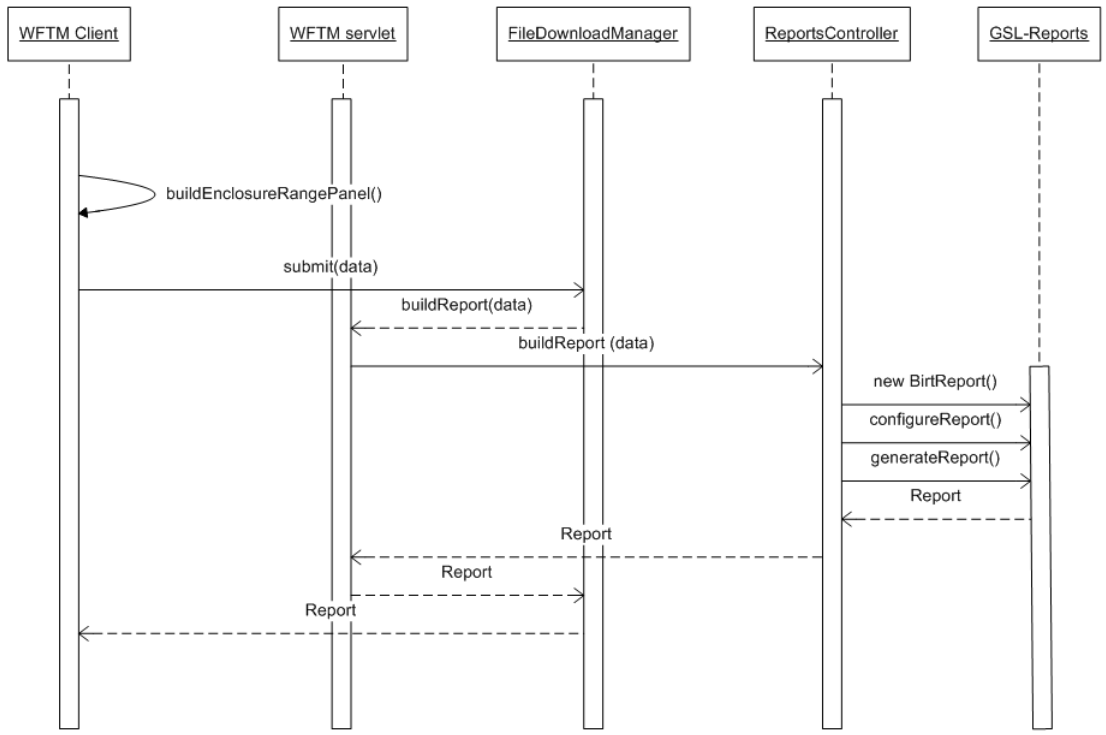

Figura C.8: *WebFTManager* - Diagrama de casos de uso (Obtener un informe)

Obtener una gráfica: El diagrama de secuencia en el caso de una gráfica (ver figura C.9) es análogo al caso de los informes, una vez que se ha pulsado el botón para la generación de una gráfica, se llama a buildChart, se selecciona el rango de fechas y la gráfica de entre las disponibles y se envían al servidor, en este caso al servidor de descargas, que se pone en comunicación con WFTM *servlet* para la generación de la gráfica, una vez que el servidor obtiene la gráfica, la devuelve al servidor de descargas y éste lo devuelve al cliente para que lo almacene en su equipo.

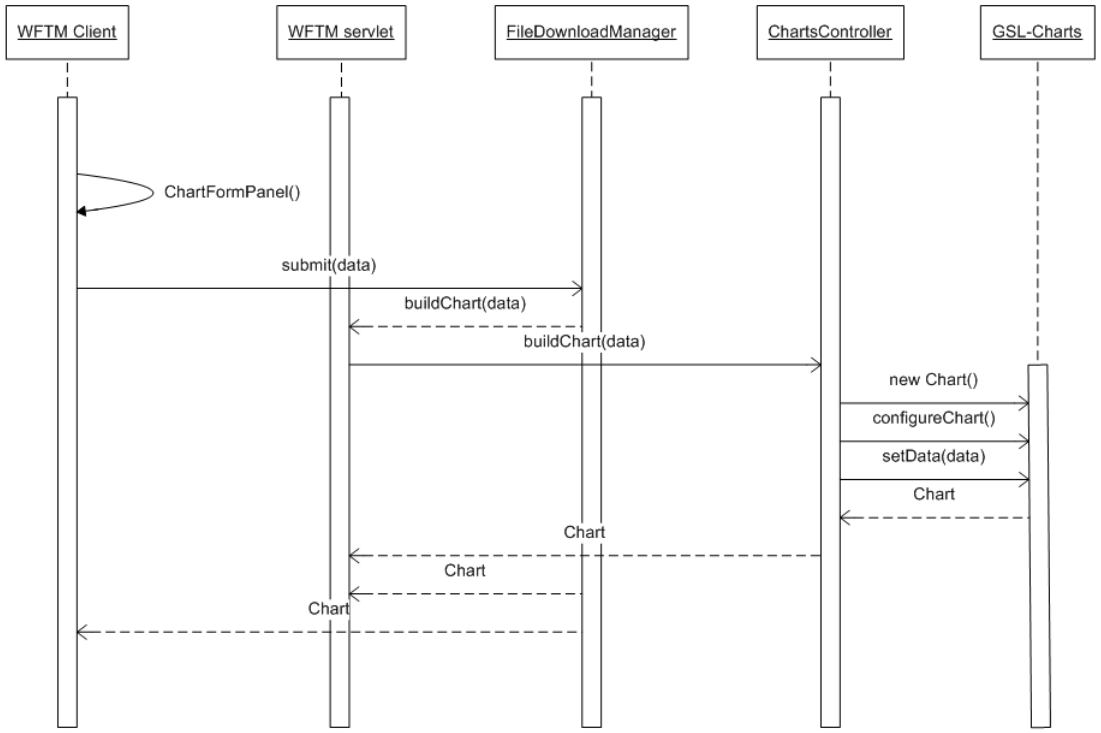

Figura C.9: *WebFTManager* - Diagrama de secuencia (Obtener una gráfica)

C.3.4. Diagrama de clases

En las figuras C.10 y C.11 se pueden apreciar los diagramas de clases de la parte cliente y servidora respectivamente de *WebFTManager*, se han expuesto en los diagramas únicamente las clases afectadas en el desarrollo de este proyecto. *WebFTManager* contiene muchas más clases que no han sido objeto de modificaciones y que incluirlas entorpecería la visualización del trabajo realizado. Dichas clases se aglutinan en sendos paquetes *iaaa.WebFTManager.client* y *iaaa.WebFTManager.server*

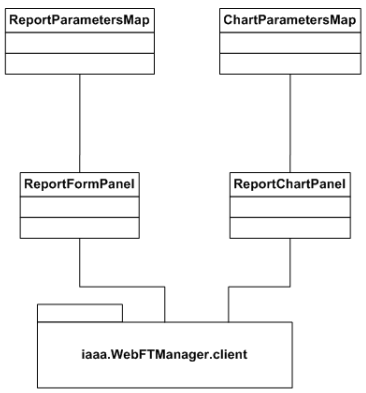

Figura C.10: *WebFTManager* - Diagrama de clases (cliente)

En el cliente ReportFormPanel y ChartFormPanel definen los paneles para la selección de los parámetros de un informe y una gráfica.

En la parte servidora el servlet ahora está ligado con *ReportController* y *ChartController* que son los controladores de los informes y de las gráficas respectivamente. Ambos controlladores se comunican con sus respectivas librerías para la generación de informes o gráficas según el caso.

FileDownloadServlet por su parte es el servidor de descargas, él se encarga de recibir las peticiones de los clientes y devolver los resultados.

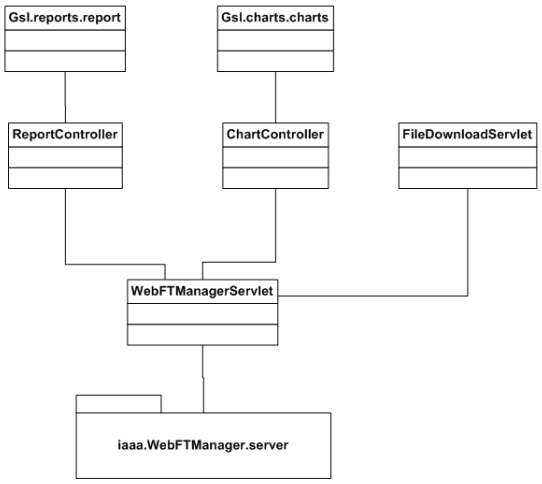

Figura C.11: *WebFTManager* - Diagrama de clases (servidor)

C.4. InfiltrationSolver

A continuación se describe con detalle el diseño de la aplicación web para el cálculo de la infiltración. Primero se citarán los requisitos funcionales y los no funcionales que debe cumplir, Seguido se mostrarán los casos de uso, los diagramas de secuencia, el diagrama de clases y finalmente el diagrama de arquitectura y el modelo de datos.

C.4.1. Requisitos

Requisitos funcionales

- Se tratará de una aplicación web que se situé como una capa por encima de *WebFTManager* capaz de realizar el cálculo de la infiltración.
- Se podrá limitar el cálculo de la infiltración con un rango de fechas.
- Durante el cálculo de la infiltración el usuario deberá conocer el estado de dicho cálculo.

Requisitos no funcionales

- La implementación de la aplicación ser hará con lenguaje Java apoyándose en el framework GWT (Google Web Toolkit).
- En caso de utilizar librerías o aplicaciones externas, la licencia de utilización debe ser libre.

C.4.2. Casos de uso

Nota: Como *InfiltrationSolver* es una especialización de *WebFTManager*, se aprovecha de todos los casos de uso de éste, los casos de uso añadidos a *WebFTManager* durante el desarrollo de este proyecto pueden verse en la sección C.3.2.

A continuación se muestran únicamente los que casos de uso exclusivos de *InfiltrationSolver* (ver figura C.12).

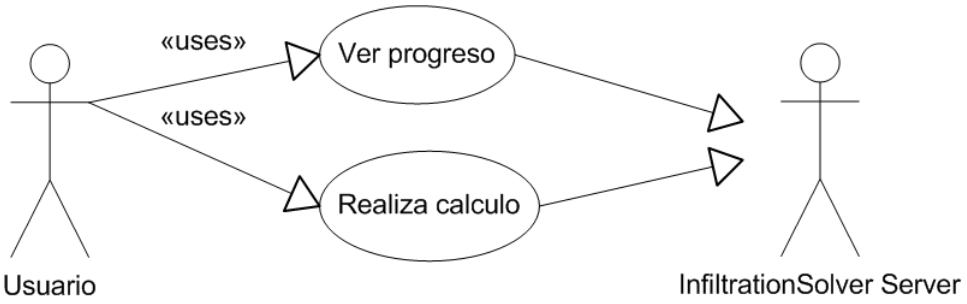

Figura C.12: *InfiltrationSolver* - Casos de uso

Calcular la infiltración: El usuario, gracias a un botón en la interfaz, lanza el cálculo de la infiltración de unos recintos y entre unas fechas previamente seleccionadas. El servidor, pone en marcha la lógica de comunicación con *PolynomialSolver*, *DecisionTableEval* para realizar los cálculos e informa del progreso al usuario.

Mostrar el progreso de los cálculos: En caso de que la ventana que muestra el estado de los cálculos se haya puesto en *background* se puede volver a mostrar el estado de los mismos.

C.4.3. Diagramas de secuencia

En la figura C.13 se puede ver la secuencia entre los diferentes actores para la realización de un cálculo de la infiltración, en la primera llamada, una vez que el *servlet* envía el cálculo al core, éste envía trabajo a los diferentes servidores de

manera autónoma. Mientras tanto el cliente consulta periódicamente al servidor el estado del cálculo al servidor.

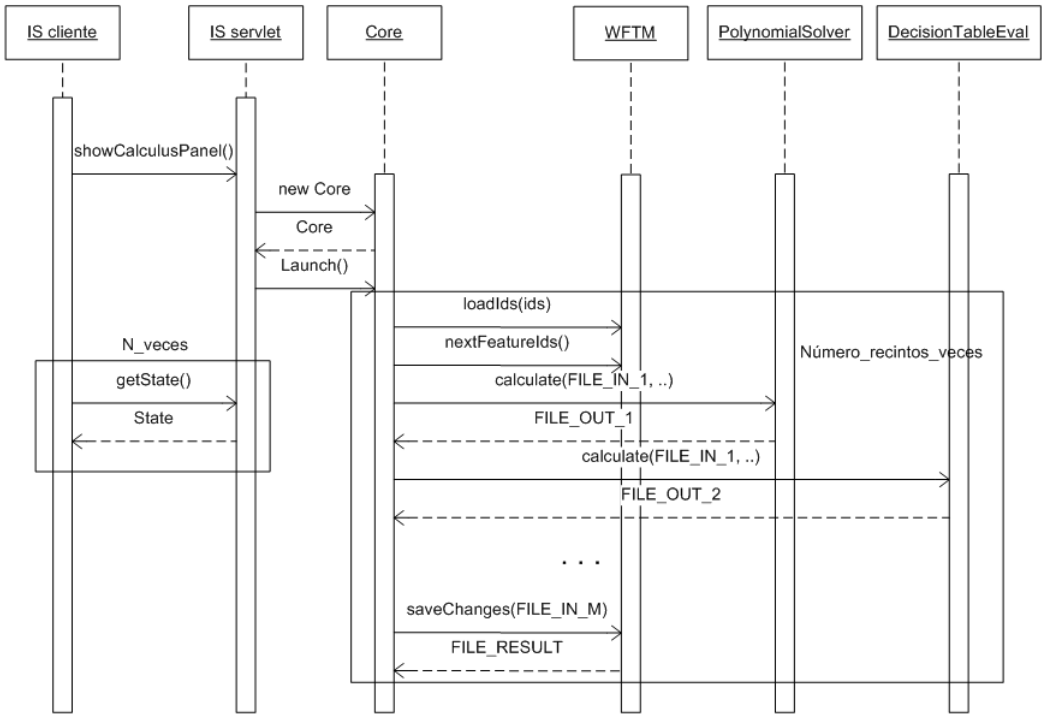

Figura C.13: Diagrama de secuencia - Realizar cálculo de infiltración

C.4.4. Diagrama de clases

En la figura C.14 y C.15 se pueden ver los diagramas de clases de *InfiltrationSolver* de la parte cliente y de la parte servidora respectivamente.

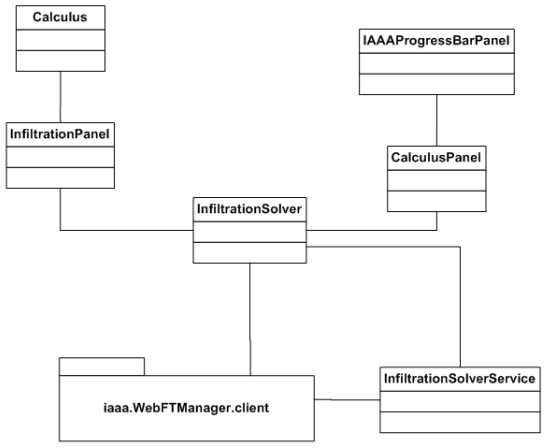

Figura C.14: *InfiltrationSolver* - Diagrama de clases parte cliente

En la parte cliente la clase *InfiltrationSolver* es la clase principal, la que tiene contacto directo con la *WebFTManager* y se encarga de la comunicación y personalización del cliente. *InfiltrationPanel* es el panel que se muestra al usuario cuando está realizando el cálculo de la infiltración que tiene como dependencia la clase calculus que define todos los atributos de un cálculo (fechas, recintos, descripción, progreso...), en la parte derecha, están las clases *CalculusPanel* que muestra los progresos de todos los cálculos.

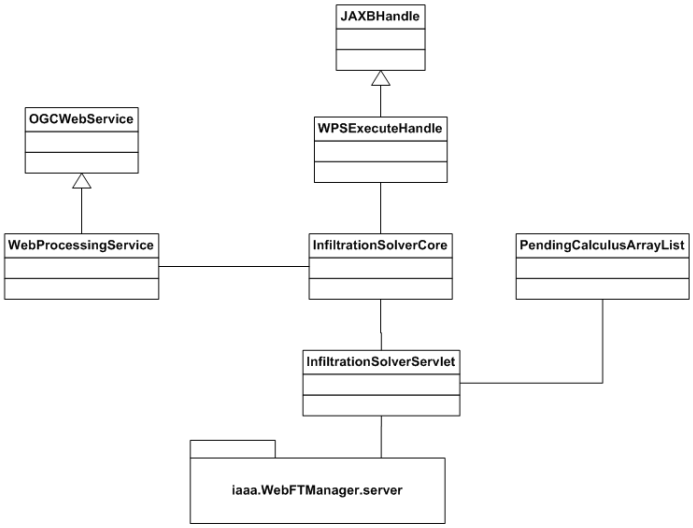

Figura C.15: *InfiltrationSolver* - Diagrama de clases parte servidora

En la parte servidora la clase principal es *InfiltrationSolverCore* que es directamente relacionado con el servlet (*InfiltrationSolver*). *InfiltrationSolverCore* se encarga de realizar los cálculos de la infiltración acuífera y por ello tiene contacto con *WebProcessingService* y *WPSExecuteHandle* ya que tiene que manejar llamadas a servidores WPS.

InfiltrationSolverServlet va ligado directamente con *PendingCalculusArrayList* que almacena la lista de cálculos pendientes en el servidor. Se puede apreciar que el servlet está relacionado con el paquete iaaa.*WebFTManager*.server y de hecho es una extensión del servlet de *WebFTManager*, esto permite acceso a los métodos de comunicación de *WebFTManager*

Anexo D

Manuales de usuario

Además de la aplicación web para el cálculo de la infiltración (*InfiltrationSolver*), se han desarrollado dos componentes más que pueden ser utilizados de manera independiente, como son la librería de informes y la librería de gráficas, éstas también tienen su correspondiente manual de usuario.

D.1. InfiltrationSolver

Tal y como muestran los casos de uso de la figura C.12, existen dos acciones principales con *InfiltrationSolver*.

Calcular la infiltración y ver el progreso. Además, como *InfiltrationSolver* hereda la funcionalidad de *WebFTManager*, también es capaz de generar informes y generar gráficas, estas dos acciones se han incluido como parte de este manual de usuario.

D.1.1. Calcular la infiltración y visualizar el progreso

Inicio

Al iniciar la aplicación se muestra la pantalla de inicio que se puede verse en la figura D.1. Esta ventana está distribuida de la siguiente manera: A la izquierda se encuentra el menú, formado por categorías (en color gris), subcategorías (en color azul oscuro) y por último opciones (en color azul claro), es desplegable y permite navegar por la aplicación fácilmente. En la parte central de la imagen, donde se ve encuentra el logotipo corporativo de GeoSpatiumLab S.L. es donde se ubica la zona de trabajo, aquí se cargará la información en función de nuestras acciones.

Paso 1: Seleccionar los parámetros

En la parte izquierda de la figura D.1 de inicio se puede observar un menú y en él, un apartado que dice "Infiltración" y justo debajo "calcular". Al pulsar en "calcular" se desplegará otro submenú con dos posibles opciones, "calcular" y "operaciones

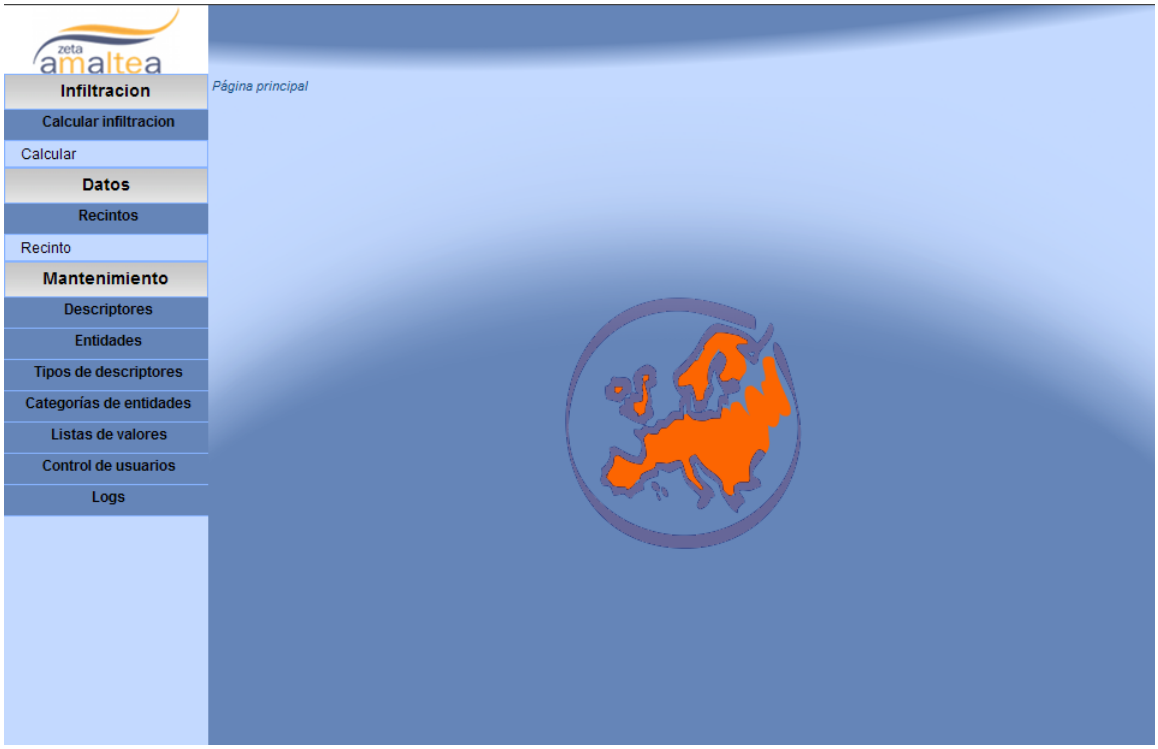

Figura D.1: Página de inicio de Infiltration Solver

en curso", si se pulsa la opción "calcular", se mostrará la ventana de selección de parámetros para calcular la infiltración, que puede verse en la figura D.2.

En dicha figura se deben seleccionar los recintos para los cuales se va a realizar el cálculo de la infiltración así como un rango de fechas para delimitar dicho cálculo. El selector de fechas de la izquierda corresponde a la fecha de inicio y el de la derecha a la fecha de finalización.

Nota: Se debe tener en cuenta que cuantos más recintos se elijan y en un rango de fechas más amplio, más costará la realización del cálculo.

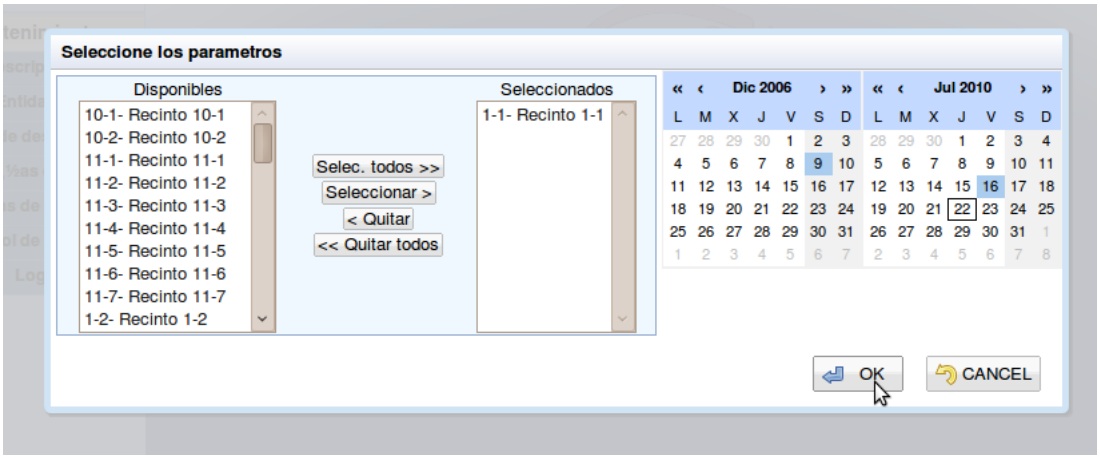

Figura D.2: Selección de parámetros para el cálculo de la infiltración

Paso 2: Visualización del progreso

Una vez seleccionados los parámetros en el paso anterior, al pulsar "OK", se ponen en marcha los mecanismos de comunicación pertinentes para resolver la infiltración acuífera y se le muestra al usuario una pantalla donde puede observar el progreso de los mismos con detalle como se ve en la figura D.3, dicha ventana además de aportar información del estado actual de los cálculos incluye una barra de progreso e informa del tiempo transcurrido y del tiempo restante estimado. Dependiendo de la cantidad de recintos y el rango de fechas que se hayan elegido en la ventana de selección de parámetros (figura D.2) los cálculos pueden durar horas, incluso días por lo que se ha habilitado un botón denominado "background" que permite esconder esta ventana y permitir al usuario seguir utilizando la aplicación con normalidad.

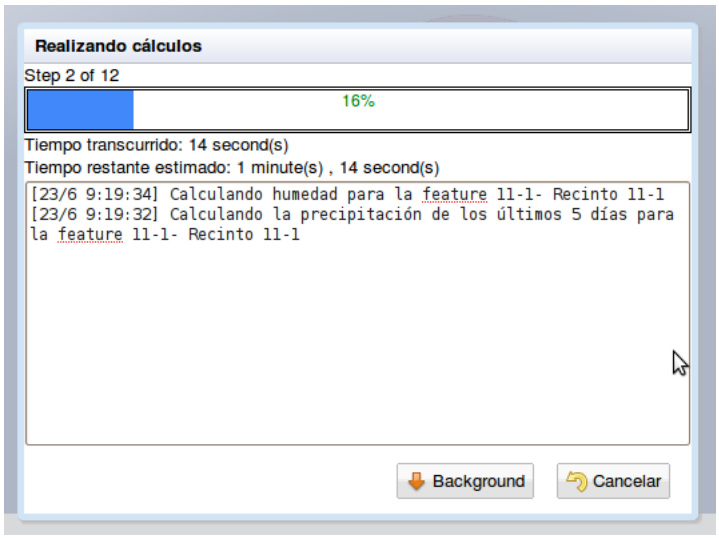

Figura D.3: Progreso de los cálculos

Paso 3: Recuperación del progreso (opcional)

Se ha contemplado el caso de que el usuario quiera, una vez escondida la ventana de progreso, comprobar nuevamente el estado de los mismos. Para ello, en el menú de la izquierda hay una opción denominada *"Operaciones en curso"* que ha sido creada con tal fin. Si mientras se está realizando un cálculo se selecciona esta opción se abrirá una pequeña ventana donde se mostrarán todas las operaciones pendientes en el servidor. Dicha ventana puede verse en la figuraD.4 .

Paso 4: Fin de los cálculos y visualización de resultados.

Una vez que los cálculos terminen, si el usuario estaba en la ventana de visualización de progreso (figura D.3) se mostrará una ventana de éxito, en caso de que los cálculos se estuviesen haciendo en "background" se enviará un mensaje al correo del usuario que hubiera comenzado los cálculos.

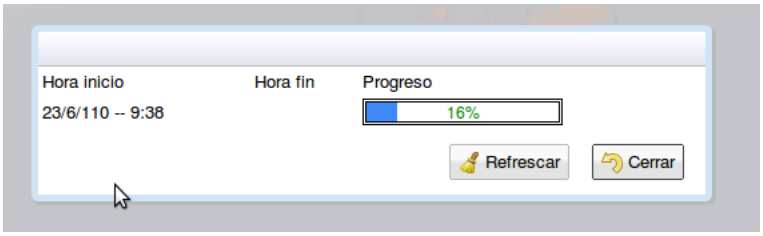

Figura D.4: Operaciones en curso

Los resultados de los cálculos corresponden a la infiltración. Éstos pueden verse en la aplicación seleccionando alguno de los recintos correspondientes y mirando su columna de infiltración, se podrá comprobar como ya dispone de valor en esa columna para los rangos de fecha seleccionados previamente. Si se desea se podrá obtener un informe o gráfica del recinto visualizado.

D.1.2. Generación de un informe

Paso 1: Abrir un recinto

Estando en la ventana de inicio, (esta ventana se explica en la sección D.1.1), al seleccionar la opción "recinto" dentro del apartado "Datos", se desplegará un submenú con las opciones "Abrir", "Nuevo", "Borrar". Si se elige la opción "Abrir", aparecerá una ventana emergente donde seleccionar un recinto a abrir (ver figura D.5).

Figura D.5: Selección de recintos

Una vez que se elija uno de ellos, se cargará en la zona de trabajo los datos del recinto así como una tabla con pestañas, que permite tener toda la información de un recinto accesible. (ver figura D.6)

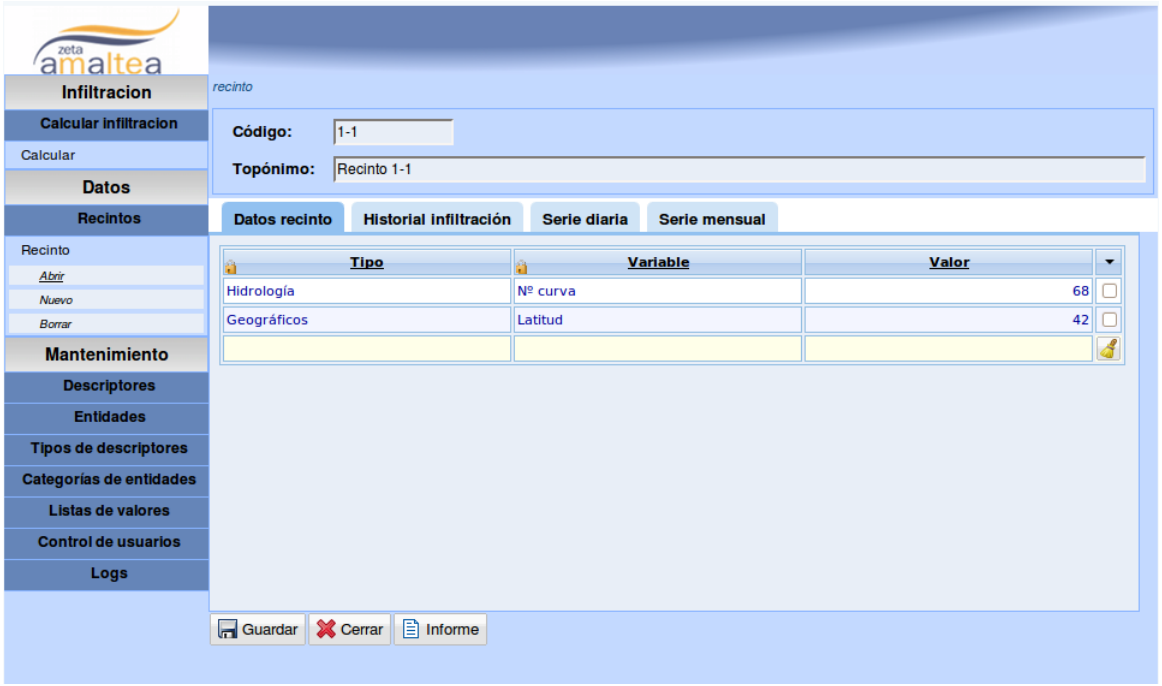

Figura D.6: Información de un recinto

Paso 2: Selección de parámetros

En la parte inferior izquierda de la zona de trabajo, se puede ver un botón con la etiqueta "Informe", si se pulsa, pasará a la ventana de selección de parámetros (ver figura D.7), en este caso se permite seleccionar el rango de fechas para el cual se requiere el informe, una vez seleccionados los días tanto de inicio como de final, pulsando en el botón "OK" el servidor comenzará a trabajar generando el informe.

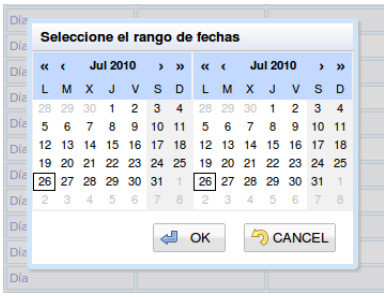

Figura D.7: Detalle de selección de parámetros

Paso 3: Guardado y visualización del informe generado

Al cabo de unos segundos, una vez que se haya generado el informe, se mostrará al usuario una ventana para que seleccione donde desea guardar el fichero resultante. (ver figura D.8)

El formato de fichero resultante es PDF y puede visualizarse fácilmente con un visor como Adobe Acrobat Reader(`http://www.adobe.com/`) o Document Viewer(`http://www.gnome.org/projects/evince`).

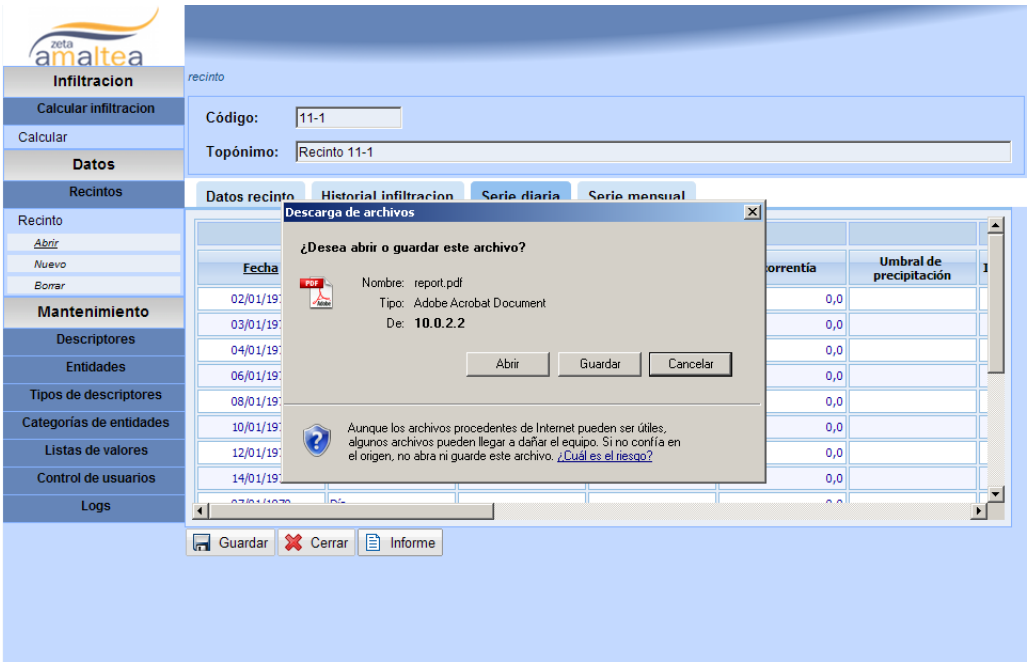

Figura D.8: Guardado de informe

D.1.3. Generación de una gráfica

Paso 1: Abrir una tabla temporal

Partiendo de la ventana de inicio (esta ventana se explica en la sección D.1.1), al navegar por la aplicación y abrir cualquier tabla de carácter temporal (con una columna de fecha y otra de precisión), en la parte inferior de la tabla aparecerá un botón permitiendo generar una gráfica puede verse un ejemplo en la figura

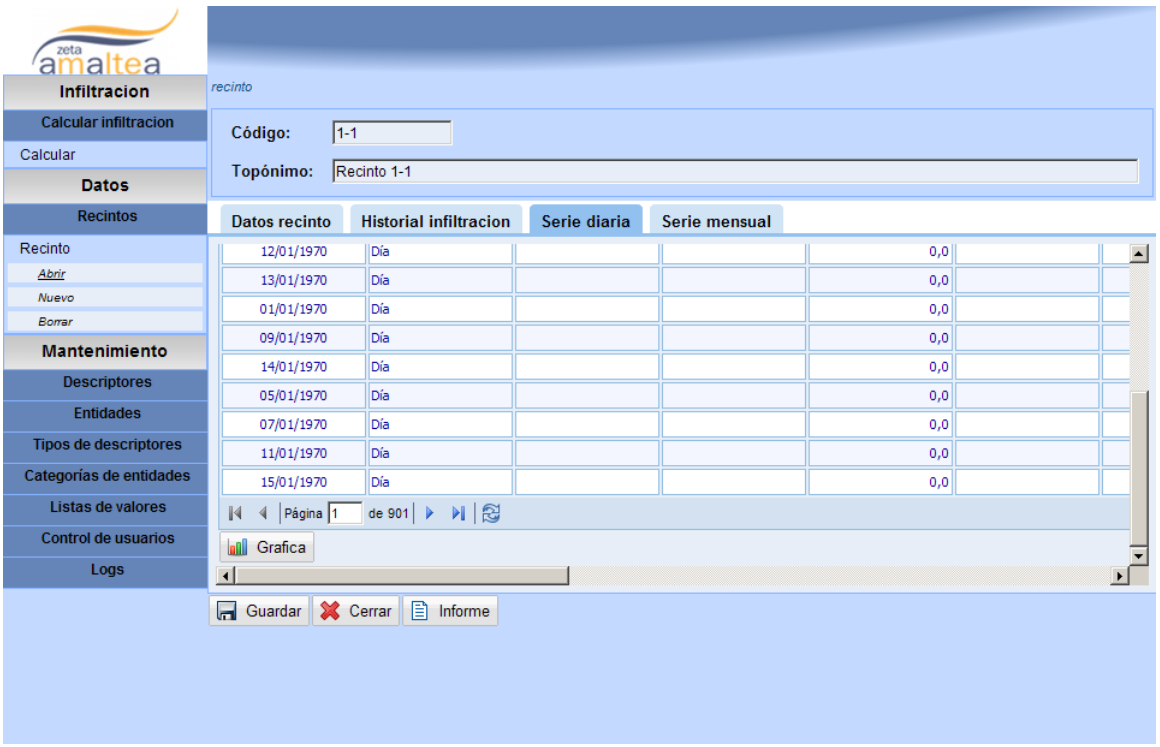

Figura D.9: Detalle de botón de gráfica

Paso 2: Seleccionando los parámetros

Al pulsar el botón de "Gráfica" se mostrará una ventana emergente permitiendo seleccionar una de las posibles gráficas para esa tabla así como el rango de fechas para las que generar dicha gráfica, puede observarse esta ventana en la figura D.10, al pulsar "OK" el servidor comenzará a trabajar generando el informe.

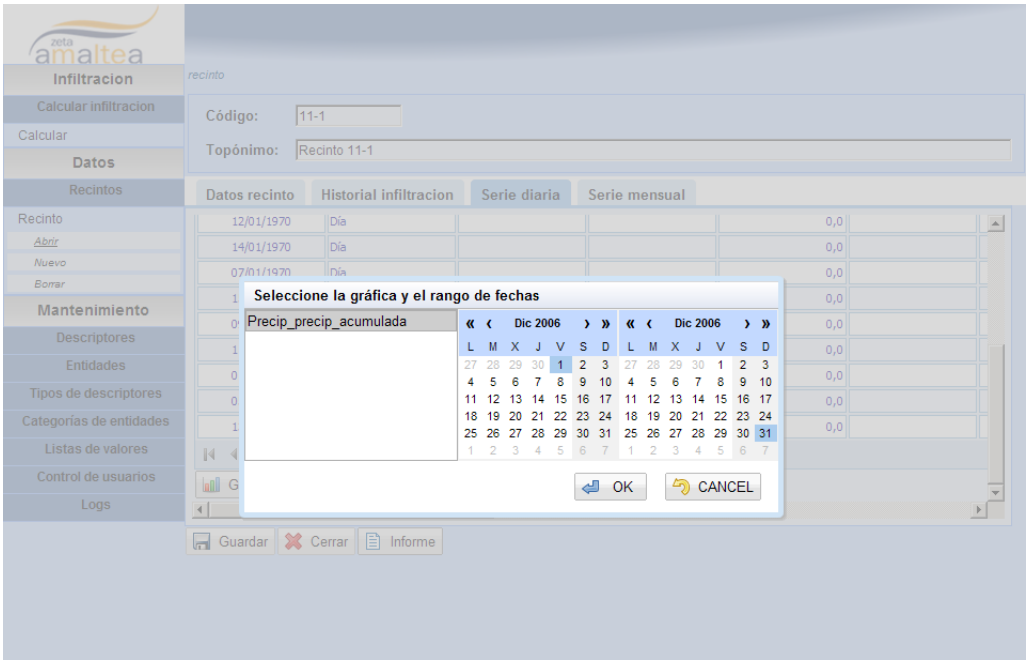

Figura D.10: Detalle de selección de paramétros para las gráficas

Paso 3: Guardado y visualización de la gráfica generada

Una vez que se haya generado la gráfica, se mostrará una ventana para que el usuario seleccione donde desea guardar la imagen JPG resultante (ver figura D.11).

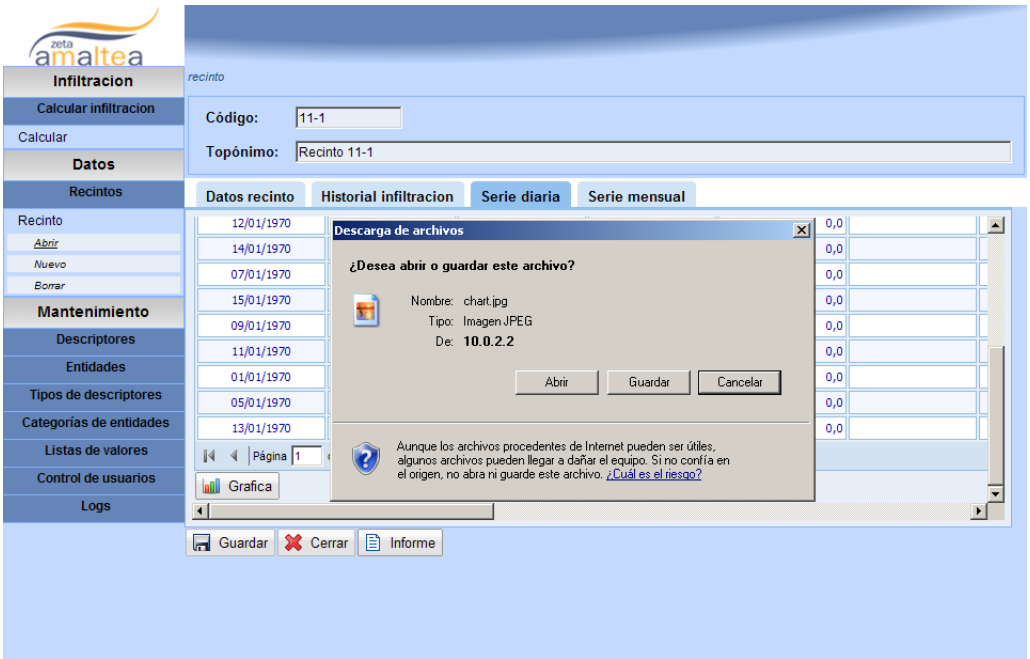

Figura D.11: Guardado de una gráfica

Cuando haya elegido el nombre y la ubicación la imagen se almacenará y estará disponible para visualizarse.

D.2. GSL-Reports

Esta librería ha sido desarrollada para permitir la personalización de una misma plantilla diseñada con BIRT desde una aplicación externa.

D.2.1. Primeros pasos

Paso1 : Instalar BIRT

Lo primero que se debe hacer es instalar BIRT en el ordenador donde se vayan a generar los informes. Descargar e instalar birt-runtime, el cual está disponible en `http://eclipse.org/birt/phoenix/`.

Paso 2: Instalar BIRT report - designer en eclipse (opcional)

Si se desea diseñar plantillas de BIRT (no es necesario si se trabaja con una plantilla ya predefinida), BIRT report-designer debe instalarse de la siguiente manera.

Para evitar problemas es muy recomendable actualizarse a la versión 3.5 de eclipse y seguir estos pasos:

1. Obtener eclipse 3.5 (o posterior) de aquí `http://www.eclipse.org/downloads/`
2. Una vez instalado navegar hasta menú Help -> Install new software
3. En la fila donde pone "Work with", introducir la siguiente dirección. `http://download.eclipse.org/birt/update-site/2.5`
4. Deseleccionar la casilla "Show Orly the latest version of new software" en caso de querer especificar una version de BIRT.
5. Asegurarse de marcar "Contact all update sites during install to find required software"
6. Elegir BIRT Framework
7. Al finalizar y dejar que eclipse resuelva las dependencias correspondientes.

Paso 3: Empezando a trabajar con *GSL-Reports*

Para empezar a trabajar con *GSL-Reports* hay que importar en su proyecto la librería *GSL-Reports*. En caso de que se quiera hacer de forma manual, con añadir a la cabecera de su fichero .java la siguiente línea es suficiente.

```
import gsl.reports.*;
```

Posteriormente es necesario configurar algunos parámetros para trabajar correctamente.

```
// birtHomeFilePath: ruta donde se encuentra instalado el birt-runtime.
String birtHomeFilePath = "/opt/birtRuntime/ReportEngine";
// resourcesPath: Ruta que BIRT utilizará para buscar los informes
//                (rptdesign), archivos css y guardar los informes
//                generados en el formato que le digamos.
String resourcesPath = "/home/user/reports/";
//Nuevo objeto BirtReport
BirtReport birtReport = new BirtReport(birtHomeFilePath, resourcesPath );
//Cambiar la plantilla
birtReport.setReportTemplate("test1");
//Para alterar las rutas de los ficheros.
//Altera la ruta donde se encuentran los ficheros CSS.
setCSSTemplatesPath(path)
setOutputPath(path)
setReportTemplatesPath(path)
```

D.2.2. Generar un informe

A partir de una plantilla de BIRT (formato .rptdesign), podemos generar un informe en diferentes formatos con la siguiente instrucción.

```
//Genera un fichero test1.xxx en la carpeta resources.
birtReport.generateReport(test1,OUTPUT_FORMAT_TYPE);
```

También es posible generar como salida un array the bytes de la siguiente manera:

```
birtReport.generateReport(outputStream, OUTPUT_FORMAT_TYPE);
```

D.2.3. Cambiar la fuente y el conjunto de datos

Las plantillas de BIRT se configuran con una fuente de datos ("data-source") y un conjunto de datos ("data-set"). *GSL-Reports* permite alterar el data-source de una plantilla, permitiendo por ejemplo obtener datos de una BBDD ORACLE y otra POSTGRES sin alterar la plantilla, o bien alterar un data-set y obtener los datos de otra tabla diferente.

```
birtReport.setDataSource(name, POSTGRESQL, host,
                         port, dataBDName, userName, password);
birtReport.setDataSet(dataSetName, "SELECT * FROM test.COD_ESTADO");
```

Entre las características más relevantes que puede realizar la librería de informes están:

D.2.4. Añadir imágenes

Se pueden añadir imágenes al informe tanto de fichero como de un array de bytes (inputStream) con la siguientes líneas.

```
// Add the image to the report from a file
birtReport.addImage(imageURL, "gsl");

// Add the image to the report from a inputStream.
birtReport.addImage(in, "MyGrid");
```

D.2.5. Eliminación y adicción de filtros en conjuntos de datos

En un mismo conjunto de datos, puede ser interesante filtrarlo para personalizar el informe.

```
// Clear all filters associated to data-set named: "oracleDataSet".
birtReport.clearFilters("oracleDataSet");

// Sets the new filter
birtReport.addEqualFilter("oracleDataSet", "ID", "3");
```

D.2.6. Paso de parámetros al informe

Aunque los filtros anteriormente explicados sirven para personalizar los datos de los informes la opción más adecuada es utilizar los parámetros, gracias a éstos se pueden desarrollar plantillas genéricas que rellenarán los datos en función de los parámetros que se le pasen, la siguiente línea contiene un ejemplo.

```
birtReport.setParameter(“feat_id”,1);
```

D.2.7. Adicción de un mapa proveniente de uno o más servidores WMS

Además de incluir imágenes (ver apartado D.2.4) se ha añadido una lógica para la inclusión de mapas provenientes de WMS.

Existen muchas variantes de adicción de mapas en el objeto birtReport, todas ellas además están duplicadas por si se quiere definir un *gridName* (lugar exacto en el informe donde se desea incluir el mapa). Lo más importante a la hora de elegir una u otra función es tener claro si nuestras peticiones (en caso de ser más de una) van a realizarse con el mismo SRS (Spatial Reference System) o no. A continuación dependiendo de sus necesidades, el subtipo de caso de uso correspondiente.

Muy importante, parámetro adaptBBOX: En las peticiones que utilizan el mismo SRS, se ofrece la posibilidad de adaptar el BBOX o no, esto significa que si tenemos un bounding box que sabemos corresponde una petición de 750x500 (ratio de aspecto 1.5:1) pero nosotros queremos la imagen en 600x600 (ratio de aspecto 1:1), la propia librería transformará el BBOX introducido para evitar la deformación de la imagen.

Nota: Se han definido 2 variables para ayudar a la legibilidad del código:

```
// Useful WMS Servers public final static
String WMS_SERVER_IDEE_BASE =
       "http://www.idee.es/wms/IDEE-Base/IDEE-Base";
String WMS_SERVER_CATASTRO =
       "http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx";
```

Petición a un solo servidor

Nota: El argumento true que aparece resaltado, corresponde con el parámetro adaptBBOX, ver apartado D.2.7.

Nota2: El argumento "-1.6,41.3576,-0.51404,42.02094" corresponde al bounding box y está formado por 4 números que definen la porción de mapa a pedir al servidor.

```
birtReport.setReportTemplate("test5");
String[] layerNames = { "Hidrografia", "transporte" };
birtReport.addMap(WMS_SERVER_IDEE_BASE,
                  new String[]{"-1.6","41.3576","-0.51404","42.02094"},
                  "EPSG:4258", 600, 600, layerNames, true);
birtReport.generateReport("test5", OUTPUT_FORMAT_TYPE.PDF);
```

Peticion a varios servidores con el mismo SRS

Pueden definirse los parámetros como ArrayList o como String[], en el siguiente ejemplo se ha utilizado String[].

```
String[] baseURLs = { WMS_SERVER_IDEE_BASE, WMS_SERVER_IDEE_BASE };
String[][] layerNamesMatrix = { { "Hidrografia" }, { "transporte" } };
birtReport.addMap(baseURLs,
                  new String[]{"-1.6","41.3576","-0.51404", "42.02094"},
WMSConstants.SRS_EPSG_4258, 600, 600, layerNamesMatrix, true);
birtReport.generateReport("report", OUTPUT_FORMAT_TYPE.PDF);
```

En el ejemplo anterior, hemos definido dos servidores (aunque sean iguales en nombre (WMS_SERVER_IDEE_BASE), realmente se tratarán de forma independiente. Además hay definida una matriz (layerNamesMatrix) de capas, donde las capas

de la primera fila de nuestra matriz serán pedidas al primer servidor (definido en baseURLs), la fila dos de nuestra matriz de capas será pedida al servidor dos, y así sucesivamente.

En este caso particular, la capa hidrología será pedida por el primer servidor (IDEE_BASE), y la capa transporte irá a la petición del segundo servidor (IDEE_BASE).

Peticion a varios servidores con distinto SRS

Aunque lo más sencillo es pedir las capas a uno o varios servidores con el mismo SRS, lo cierto es que a veces no es posible por incompatibilidad de éstos. La librería ofrece la posibilidad de definir nuestras peticiones como nosotros queramos aunque con ciertas limitaciones.

Muy importante: La librería en este caso, únicamente hará las peticiones "tal cual" le vengan del cliente, por lo que será responsabilidad del cliente conocer perfectamente los bounding box correspondientes de cada petición para que, al fusionar las imágenes procedentes de los distintos servidores, éstas encajen perfectamente.

Nota: El parámetro adaptBBOX deberá ponerse a false ya que no es posible ajustar BBOX de varios SRS si no es haciendo transformación de coordenadas.

```
String[] ideBaselayerNames = { "Hidrografia", "transporte" };
WMSRequestType wmsIDEEBASE = new WMSRequestType(WMS_SERVER_IDEE_BASE,
                             new String[]{"675681.0844595262",
                                          "4613932.319103513",
                                          "677876.3885705681",
                                          "4615884.063828118"},
                             WMSConstants.SRS_EPSG_23030, 750,
                             500, ideBaselayerNames, false);
String[] catastroLayerNames = { "catastro" };
WMSRequestType wmsCatastro = new WMSRequestType(WMS_SERVER_CATASTRO,
                             new String[]{"-0.8898643970045188",
                                          "41.65680636512691",
                                          "-0.8634976645435641",
                                          "41.67438415664217"},
                             "EPSG%3A4230", 750,500,
                             catastroLayerNames, false);
ArrayList<WMSRequestType> wmsRequests = new ArrayList<WMSRequestType>();
wmsRequests.add(wmsIDEEBASE);
wmsRequests.add(wmsCatastro);
birtReport.addMap(wmsRequests, "myGrid");
birtReport.generateReport("report", OUTPUT_FORMAT_TYPE.PDF);
```

Petición a uno o varios servidores pero incluyendo características SLD

```
String SLD1 = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
               <ogc:GetMap
               ...
               </ogc:GetMap>";
String baseURL1 = "http://www.idee.es/BCN2000-OWS/ogcwebservice"; ç
String SLD2 = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
              <ogc:GetMap
              ...
              </ogc:GetMap>";
String baseURL2 = "http://www.idee.es/BCN2000-OWS/ogcwebservice";
String[] SLDs = { SLD1, SLD2 };
String[] baseURLs = { baseURL1, baseURL2 };
birtReport.addMap(baseURLs, SLDs, null);
birtReport.generateReport("report", OUTPUT_FORMAT_TYPE.PDF);
```

Nota: Los Strings SLD corresponden con el contenido de los ficheros XML.

Para ilustrar el ejemplo simplemente se ha puesto el comienzo y final de esos ficheros en los Strings SLD.

Muy importante: El SLD es tratado como un String en la librería así que debe asegurarse que cadenas como está:

xml version="1.0" encoding . . . son transformadas por esta otra xml version=\"1.0\" encoding . . . para que java entienda el String correctamente.

Aplicar un estilo a partir de un fichero CSS

Cuando se crea una plantilla de BIRT (fichero con extensión rptdesign), se puede darle estilo a todos los componentes, pero si se quiere modificar el aspecto de su informe, solo hay que hacer:

birtReport.setCSSTemplate(cssTemplateName);

birtReport.generateReport("report", OUTPUT_FORMAT_TYPE.PDF);

Muy importante: Los estilos que existan en el rptdesign serán eliminados y sobreescritos por el nuevo fichero de estilos.

D.3. GSL-Charts

D.3.1. Antes de empezar

Paso1 : Instalar R

Para poder hacer funcionar *GSL-Charts* es necesario tener instalado el paquete matemático R que se encargará de realizar las gráficas.

Se puede descargar R desde su página web oficial `http://www.r-project.org/` teniendo en cuenta la versión de su sistema operativo. Para garantizar la máxima compatibilidad con la librería se recomienda obtener la versión con la que ha sido probada (2.9.2). También hay que descargarse la versión 2.5-1 del paquete XML de la siguiente dirección `http://cran.es.r-project.org/web/packages/XML/index.html` e instalarla como especifica el README. No obstante si se trabaja en Ubuntu, la forma más sencilla de instalar R es ejecutar en una consola:

```
sudo aptitude install r-base
sudo aptitude install r-base-dev
sudo aptitude install r-cran-xml
```

Nota: Actualmente Ubuntu incorpora nativamente R en sus repositorios, de no ser así, en la página oficial de r-project `http://www.r-project.org/`, están disponibles los repositorios oficiales con las últimas versiones de R.

Paso 2: Obtener una gráfica en formato XML

Una vez que hayamos instalado satisfactoriamente R en nuestro ordenador (ver apartado D.3.1) necesitamos un fichero XML que defina una gráfica para poder decirle a *GSL-Charts* qué tipo de gráfica queremos. Para obtener más información sobre el formato de las gráficas ver sección D.3.4.

D.3.2. Rellenar de datos una gráfica (opcional)

Si el fichero XML no contiene los datos a representar en la gráfica, éstos deben pasarse como tipo ChartData. El objeto ChartData contiene 2 atributos:

```
    // Nombre de las series.
private String[] serieId;

// Map donde para cada componente X, tenemos uno o varios valores de Y
private Map<String, String[]> map;
```

Este objeto lo podemos rellenar fácilmente con una función como la siguiente:

```
private ChartData FillData() {
        ChartData chartData = new ChartData(3);
        chartData.setSerieId(0, "X");
        chartData.setSerieId(1, "LENGTH");
        chartData.setSerieId(2, "LAT");

        for (int i=0; i<40; i++){
            String[] map = new String[3];
            map[0] = String.valueOf(i);
            map[1] = String.valueOf(i*2);
            map[2] = String.valueOf(i*i);
            chartData.putMap(String.valueOf(i), map);
        }
        return chartData;
    }
```

Nota: Para que luego no existan incompatibilidades, las series definidas en el objeto ChartData (X, LENGHT y LAT en este caso) deben estar presentes en el fichero XML que define la grafica.

D.3.3. Generar una gráfica

```
//Ruta donde está instalado R
public final static String rBinarie = "/usr/lib/R/bin/R";

//Ruta que utilizará R para trabajar
public final static String _resourcesPath = "/home/user/gsl_charts/output";
public void test(){
try{
//Fichero de configuración de una gráfica (creado previamente)
    String _ConfigXMLFile = "/home/user/gsl_charts/input/simple.xml";

//Función para rellenar un objeto de tipo ChartData
    ChartData chartData = FillData();

// Creamos un nuevo objeto de tipo Chart.
    Chart chart = new Chart(rBinarie, resourcesPath);

// Aplicamos la configuración con el fichero XML creado previamente
    chart.setXMLConfigurationFileName(_ConfigXMLFile);

// Aplicamos los datos que queremos representar.
    chart.setData(chartData);

// Por último generamos la gráfica.
    chart.generateChart();


// Una vez que hemos generado la gráfica tenemos a nuestra
// disposición un objeto de tipo OutputStream que podemos obtener
```

```
// con la función getImage().
    File file = new File(resourcesPath + "/gráfica.jpg");
    ImageIO.write(chart.getImage(), chart.getImageFormat(), file);
}catch (Exception e){
    System.err.println(“Error en la gráfica: + e.printStackTrace()”);
}
```

D.3.4. Formato de gráfica

Si queremos crear una nueva gráfica o modificar alguna existente tendremos que ajustarnos a los siguientes XML schemas

GetChart.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:wchs="http://iaaa.cps.unizar.es/wchs"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:jxb="http://java.sun.com/xml/ns/jaxb" jxb:version="1.0"
  targetNamespace="http://iaaa.cps.unizar.es/wchs">
  <xsd:annotation>
    <xsd:appinfo>
      <jxb:schemaBindings>
        <jxb:package name="iaaa.jaxb.wchs"/>
      </jxb:schemaBindings>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:include schemaLocation="wchs.xsd"></xsd:include>
  <xsd:include schemaLocation="Chart.xsd"></xsd:include>

  <xsd:element name="ChartDefinition" type="wchs:ChartDefinitionType"
       abstract="true"/>
  <xsd:complexType abstract="true" name="ChartDefinitionType" />
  <xsd:element name="UserDefinedChart" type="wchs:UserDefinedChartType"
       substitutionGroup="wchs:ChartDefinition" />
  <xsd:complexType name="UserDefinedChartType" >
    <xsd:complexContent>
      <xsd:extension base="wchs:ChartDefinitionType" >
        <xsd:sequence>
          <xsd:element ref="wchs:Chart" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="NamedChart" type="wchs:NamedChartType"
       substitutionGroup="wchs:ChartDefinition" />
  <xsd:complexType name="NamedChartType" >
    <xsd:complexContent>
      <xsd:extension base="wchs:ChartDefinitionType" >
        <xsd:sequence>
          <xsd:element ref="wchs:Chart" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
```

```
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Data" type="wchs:DataType" />
<xsd:complexType name="DataType" >
  <xsd:choice>
    <xsd:element name="url" type="xsd:anyURI" />
    <xsd:element name="cdata" type="xsd:string" />
  </xsd:choice>
</xsd:complexType>

<xsd:element name="MappingField" type="wchs:MappingFieldType" />
<xsd:complexType name="MappingFieldType">
  <xsd:sequence>
    <xsd:element name="operation" minOccurs="0" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="mean"></xsd:enumeration>
          <xsd:enumeration value="max"></xsd:enumeration>
          <xsd:enumeration value="min"></xsd:enumeration>
          <xsd:enumeration value="sum"></xsd:enumeration>
          <xsd:enumeration value="prod"></xsd:enumeration>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="dataField" type="xsd:string" use="required"/>
  <xsd:attribute name="serieId" type="xsd:string" use="required"/>
  <xsd:attribute name="serieAxis" type="wchs:SerieAxisType" use="required"/>
  <xsd:attribute name="dataType" type="xsd:string" use="required"/>
 <xsd:attribute name="dataFormat" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:element name="SerieAxis" type= "wchs:SerieAxisType"/>
<xsd:simpleType name="SerieAxisType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="x|y"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="Mapping" type="wchs:MappingType" />
<xsd:complexType name="MappingType">
  <xsd:sequence>
    <xsd:element ref="wchs:MappingField" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="natural">
  <xsd:restriction base="xsd:int">
    <xsd:minExclusive value="0"></xsd:minExclusive>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="SizeType">
  <xsd:sequence>
    <xsd:element name="Width" type="wchs:natural" minOccurs="1"
         maxOccurs="1" />
    <xsd:element name="Height" type="wchs:natural" minOccurs="1"
         maxOccurs="1"/>
  </xsd:sequence>
```

```
  </xsd:complexType>

  <xsd:element name="Output" type="wchs:OutputType" />
  <xsd:complexType name="OutputType" >
    <xsd:sequence>
      <xsd:element name="Format" minOccurs="1" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="image/jpeg"></xsd:enumeration>
            <xsd:enumeration value="image/png"></xsd:enumeration>
            <xsd:enumeration value="image/bmp"></xsd:enumeration>
            <xsd:enumeration value="application/pdf"></xsd:enumeration>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="Size" type="wchs:SizeType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="GetChart" type="wchs:GetChartType" />
  <xsd:complexType name="GetChartType" >
    <xsd:complexContent>
      <xsd:extension base="wchs:BaseRequestType">
      <xsd:all>
        <xsd:element ref="wchs:ChartDefinition" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="wchs:Data" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="wchs:Mapping" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="wchs:Output" minOccurs="1" maxOccurs="1" />
      </xsd:all>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Chart.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:wchs="http://iaaa.cps.unizar.es/wchs"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:jxb="http://java.sun.com/xml/ns/jaxb" jxb:version="1.0"
  targetNamespace="http://iaaa.cps.unizar.es/wchs">
  <xsd:element name="Format" type="wchs:FormatType"/>
  <xsd:complexType name="FormatType">
    <xsd:attribute name="color" type="xsd:string" use="optional"/>
    <xsd:attribute name="size" type="xsd:float" use="optional"/>
  </xsd:complexType>

  <xsd:element name="LineFormat" type="wchs:LineFormatType"
       substitutionGroup="wchs:Format" />
  <xsd:complexType name="LineFormatType">
    <xsd:complexContent>
      <xsd:extension base="wchs:FormatType" >
        <xsd:attribute name="lineStyle" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="PointFormat" type="wchs:PointFormatType"
       substitutionGroup="wchs:Format" />
  <xsd:complexType name="PointFormatType">
    <xsd:complexContent>
      <xsd:extension base="wchs:FormatType" >
        <xsd:attribute name="pointStyle" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="BarFormat" type="wchs:BarFormatType"
       substitutionGroup="wchs:Format" />
  <xsd:complexType name="BarFormatType">
    <xsd:complexContent>
      <xsd:extension base="wchs:FormatType" >
        <xsd:attribute name="borderColor" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="LetterFormat" type="wchs:LetterFormatType"
       substitutionGroup="wchs:Format" />
  <xsd:complexType name="LetterFormatType">
    <xsd:complexContent>
      <xsd:extension base="wchs:FormatType" >
        <xsd:attribute name="family" type="xsd:string" use="optional"/>
        <xsd:attribute name="font" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Representation" type="wchs:RepresentationType"
       abstract="true"/>
  <xsd:complexType abstract="true" name="RepresentationType" />
  <xsd:element name="Line" type="wchs:LineType"
       substitutionGroup="wchs:Representation" />
  <xsd:complexType name="LineType">
```

```
  <xsd:complexContent>
    <xsd:extension base="wchs:RepresentationType" >
      <xsd:sequence>
        <xsd:element ref="wchs:LineFormat" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Point" type="wchs:PointType"
     substitutionGroup="wchs:Representation" />
<xsd:complexType name="PointType">
  <xsd:complexContent>
    <xsd:extension base="wchs:RepresentationType" >
      <xsd:sequence>
        <xsd:element ref="wchs:PointFormat" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Bar" type="wchs:BarType"
     substitutionGroup="wchs:Representation" />
<xsd:complexType name="BarType">
  <xsd:complexContent>
    <xsd:extension base="wchs:RepresentationType" >
      <xsd:sequence>
        <xsd:element ref="wchs:BarFormat" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="type" minOccurs="0" maxOccurs="1">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="normal"></xsd:enumeration>
              <xsd:enumeration value="stack"></xsd:enumeration>
              <xsd:enumeration value="percentage"></xsd:enumeration>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Serie" type="wchs:SerieType" />
<xsd:complexType name="SerieType">
  <xsd:sequence>
    <xsd:element ref="wchs:Representation" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:element name="Box" type="wchs:BoxType"/>
<xsd:complexType name="BoxType">
  <xsd:all>
    <xsd:element ref="wchs:LineFormat" minOccurs="0" maxOccurs="1"/>
  </xsd:all>
</xsd:complexType>
<xsd:element name="ScaleElement" type="wchs:ScaleElementType" abstract="true"/>
<xsd:complexType name="ScaleElementType" abstract="true">
```

```
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="Start" type="wchs:StartType"
     substitutionGroup="wchs:ScaleElement"/>
<xsd:complexType name="StartType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ScaleElementType" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="End" type="wchs:EndType"
     substitutionGroup="wchs:ScaleElement"/>
<xsd:complexType name="EndType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ScaleElementType" />
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Interval" type="wchs:IntervalType"
     substitutionGroup="wchs:ScaleElement"/>
<xsd:complexType name="IntervalType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ScaleElementType" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="IntervalScale" type="wchs:IntervalScaleType"
     substitutionGroup="wchs:Scale"/>
<xsd:complexType name="IntervalScaleType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ScaleType">
      <xsd:sequence>
        <xsd:element ref="wchs:Start" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="wchs:End" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="wchs:Interval" minOccurs="1" maxOccurs="1"/>

      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Element" type="wchs:ElementType"
     substitutionGroup="wchs:ScaleElement"/>
<xsd:complexType name="ElementType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ScaleElementType" />
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DiscreteScale" type="wchs:DiscreteScaleType"
     substitutionGroup="wchs:Scale"/>
<xsd:complexType name="DiscreteScaleType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ScaleType">
      <xsd:sequence>
        <xsd:element ref="wchs:Element" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
```

```
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Scale" type="wchs:ScaleType" abstract="true" />
<xsd:complexType abstract="true" name="ScaleType">
  <xsd:attribute name="valueType" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:element name="Axis" type="wchs:AxisType" abstract="true"/>
<xsd:complexType name="AxisType" abstract="true">
  <xsd:sequence>
    <xsd:element ref="wchs:Title" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wchs:Scale" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="mark" type="xsd:boolean" use="optional"/>
  <xsd:attribute name="namesInMarks" type="xsd:boolean" use="optional"/>
  <xsd:attribute name="markSense" type="xsd:int" use="optional"/>
</xsd:complexType>
<xsd:element name="YAxis" type="wchs:YAxisType" substitutionGroup="wchs:Axis"/>
<xsd:complexType name="YAxisType">
  <xsd:complexContent>
    <xsd:extension base="wchs:AxisType">
      <xsd:sequence>
        <xsd:element name="serie" type="xsd:string" minOccurs="0"
             maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="right" type="xsd:boolean" use="optional"
           default="false"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="XAxis" type="wchs:XAxisType" substitutionGroup="wchs:Axis"/>
<xsd:complexType name="XAxisType">
  <xsd:complexContent>
    <xsd:extension base="wchs:AxisType">
      <xsd:attribute name="top" type="xsd:boolean" use="optional"
           default="false"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="Title" type="wchs:TitleType"/>
<xsd:complexType name="TitleType">
  <xsd:all>
    <xsd:element ref="wchs:LetterFormat" minOccurs="0" maxOccurs="1"/>
  </xsd:all>
<xsd:attribute name="label" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:element name="Legend" type="wchs:LegendType"/>
<xsd:complexType name="LegendType">
  <xsd:sequence>
    <xsd:element ref="wchs:Title" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wchs:Box" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="location" type="xsd:string" use="required"/>
  <xsd:attribute name="ncol" use="optional">
```

```
    <xsd:simpleType>
      <xsd:restriction base="xsd:int">
        <xsd:minInclusive value="1"></xsd:minInclusive>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="size" type="xsd:float" use="optional"/>
</xsd:complexType>
<xsd:element name="Grid" type="wchs:GridType"/>
<xsd:complexType name="GridType">
  <xsd:all>
    <xsd:element ref="wchs:LineFormat" minOccurs="0" maxOccurs="1"/>
  </xsd:all>
  <xsd:attribute name="nx" type="xsd:int" use="required" />
  <xsd:attribute name="ny" type="xsd:int" use="required" />
</xsd:complexType>

<xsd:element name="Chart" type="wchs:ChartType" abstract="true" />
<xsd:complexType abstract="true" name="ChartType">
  <xsd:sequence>
    <xsd:element name="Box" type="wchs:BoxType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="Title" type="wchs:TitleType" minOccurs="0"
         maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="background" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:element name="AxisChart" type="wchs:AxisChartType"
     substitutionGroup="wchs:Chart"/>
<xsd:complexType name="AxisChartType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ChartType">
      <xsd:sequence>
        <xsd:element name="YAxis" type="wchs:YAxisType" minOccurs="0"
             maxOccurs="2"/>
        <xsd:element name="XAxis" type="wchs:XAxisType" minOccurs="0"
             maxOccurs="1"/>
        <xsd:element name="Serie" type="wchs:SerieType" minOccurs="0"
             maxOccurs="unbounded"/>
        <xsd:element name="Legend" type="wchs:LegendType" minOccurs="0"
             maxOccurs="1"/>
        <xsd:element name="Grid" type="wchs:GridType" minOccurs="0"
             maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="PieChart" type="wchs:PieChartType"
     substitutionGroup="wchs:Chart"/>
<xsd:complexType name="PieChartType">
  <xsd:complexContent>
    <xsd:extension base="wchs:ChartType">
      <xsd:sequence>
        <xsd:element name="color" type="xsd:string" minOccurs="0"
             maxOccurs="unbounded"/>
```

```
        </xsd:sequence>
        <xsd:attribute name="serieId" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Ejemplo de gráfica

Un ejemplo válido podría ser este:

```
<?xml version="1.0" encoding="UTF-8"?>
<wchs:GetChart xmlns:wchs="99"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="99 ../schemas/GetChart.xsd ">
     <wchs:Output>
      <Format>image/jpeg</Format>
             <Size>
             <Width>900</Width>
             <Height>900</Height>
             </Size>
  </wchs:Output>
  <wchs:UserDefinedChart>
     <wchs:AxisChart background="230,240,250">
        <Serie id="LENGTH">
           <wchs:Line>
              <wchs:LineFormat color="255,0,0" size="1" lineStyle="solid"/>
           </wchs:Line>
        </Serie>
        <Serie id="LAT">
           <wchs:Point>
              <wchs:PointFormat color="0,0,255" size="1"
                    pointStyle="fill_circle"/>
           </wchs:Point>
        </Serie>
        <Legend location="bottom"/>
        <Box />
        <Title label="Titulo de la grafica"></Title>
     </wchs:AxisChart>
  </wchs:UserDefinedChart>
  <wchs:Data>
  </wchs:Data>
  <wchs:Mapping>
     <wchs:MappingField dataField="X" serieAxis="x"
                        serieId="X" dataType="string"/>
     <wchs:MappingField dataField="LENGTH" serieAxis="y"
                        serieId="LENGTH" dataType="int" />
     <wchs:MappingField dataField="LAT" serieAxis="y"
                        serieId="LAT" dataType="int" />
  </wchs:Mapping>
</wchs:GetChart>
```

Anexo E

Cálculo de la infiltración

El método que ha sido utilizado para realizar el cálculo de la infiltración acuífera y que se presenta en este capítulo es el denominado "método del número de curva".

E.1. Definiciones previas

Escorrentía: Es la lámina de agua que circula sobre la superficie en una cuenca de drenaje, es decir la altura en milímetros del agua de lluvia escurrida y extendida.

Precipitación: Caída de agua sólida o líquida por la condensación del vapor sobre la superficie terrestre.

Infiltración: Cantidad de agua que penetra en el suelo.

Evotranspiración: La evotranspiración o transpiración vegetal es el proceso por el cual las plantas devuelven parte del agua que consumen en forma de vapor de agua.

Número de curva: El número de curva es un parámetro empírico (entre 0 y 100) que relaciona la escorrentía en función del tipo y condición de suelo. Los tipos de suelo con baja infiltración tendrán un número de curva alto y viceversa.

E.2. Cálculo teórico

La **infiltración** (Inf) se calcula como la precipitación menos la escorrentía (P-Esc), aunque hay que tener en cuenta ciertos matices que se detallan a continuación.

La **precipitación** (P) comienza a producir escorrentía (Esc) cuando supera el valor umbral de precipitación (I_o), $I_o = 0{,}2 \cdot$ S La diferencia P-Esc tiende a un valor constante, que representa la infiltración máxima (S), expresado en mm. Para la obtención del valor de S se utiliza la siguiente expresión , donde NC es el número de curva:

$$S = 254 \left[\frac{100}{NC} - 1\right]$$

La **escorrentía** generada por un evento de precipitación P se calcula mediante:

$$Esc = \frac{(P-0{,}2)^2}{(P+0{,}8S)}$$

Se obtiene así una serie diaria de escorrentía superficial generada en función de su número de curva.

El **estado de humedad del suelo** previo al episodio de lluvia condiciona la capacidad de infiltración de éste y, por tanto, hay que aplicar una corrección al número de curva de la cuenca en función de estas condiciones de humedad precedentes.

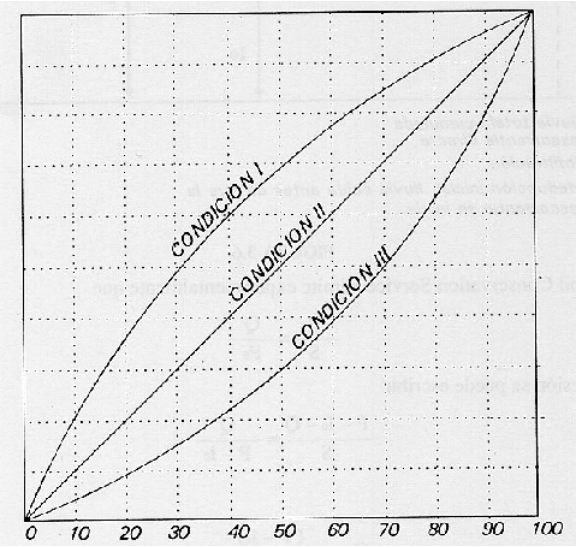

Figura E.1: Corrección del número de curva

Se establecen así tres situaciones (ver figura E.1) :

- Si durante la precipitación el suelo se encuentra seco la escorrentía será menor (Condición I).
- Si por el contrario el suelo se encuentra saturado debido a las lluvias precedentes la escorrentía será mayor (Condición III);
- Las condiciones de humedad media quedan incluidas en la Condición II.

A este respecto, para determinar las condiciones de humedad del suelo en el momento de la lluvia y el tipo de corrección a aplicar, se considera la precipitación acumulada los cinco días precedentes (P5) según se indica en el en la tabla E.1 (TRAGSA, 1998):

La fórmula para calcular P5 es: $P5 = \sum_{i-6}^{i-1} P$

Para calcular la **Evapotranspiración potencial** (ETP) se utiliza la fórmula de Thornthwaite que consiste en:

1. Calcular un índice de calor mensual (i) a partir de la temperatura media mensual (t)

Condición	Periodo húmedo (Oct -Mar)	Periodo seco (Abr - Sep)
I	P5 $<$ 12.5 mm	P5 $<$ 35.5 mm
II	12.5 $<$ P5 $<$ 28 mm	35.5 $<$ P5 $<$ 53 mm
III	P5 $>$ 28	P5 $>$ 53 mm

Tabla E.1: Condiciones de humedad

$$i = \left(\frac{t}{5}\right)^{1,514}$$

2. Se calcula el índice de calor anual (I) como la suma de los i anteriormente calculados:

$$I = \sum i$$

3. Se calcula el ETP mensual "sin corregir" mediante la fórmula:

$$ETP_{sincorregir} = 16\left(\frac{10t}{I}\right)^{a}$$

Donde $ETP_{sincorregir}$ = ETP mensual en mm/mes para meses de 30 días y 12 horas de sol (teóricas)

t = temperatura media mensual en ^{o}C

I = índice de calor anual

$a = 675 \cdot 10^{-9} I^3 - 771 \cdot 10^{-7} I^2 + 1792 \cdot 10^{-5} I + 0,49239$

4. Correción para el número de días del mes y el número de horas de sol:

$$ETP_{corregida} = ETP_{sincorregir} \frac{N}{12} \frac{d}{30}$$

Donde:

$ETP_{corregida}$ = Evotranspiración potencial corregida

N = número máximo de horas de sol dependiendo del mes y la latitud

d = número de días del mes

Para el cálculo de la **infiltración** se van a aplicar dos algoritmos, éstos pueden en la figura E.2 y en las tablas E.2 y E.3

El calculo de la infiltración con el algoritmo 1:

Condición de humedad	Cálculo de Inf
Condición III	Inf = P-Esc
Condición II	Inf = 0
Condición I	Inf = 0

Tabla E.2: Algoritmo 1 para calcular la infiltración

El calculo de la infiltración con el algoritmo 2:

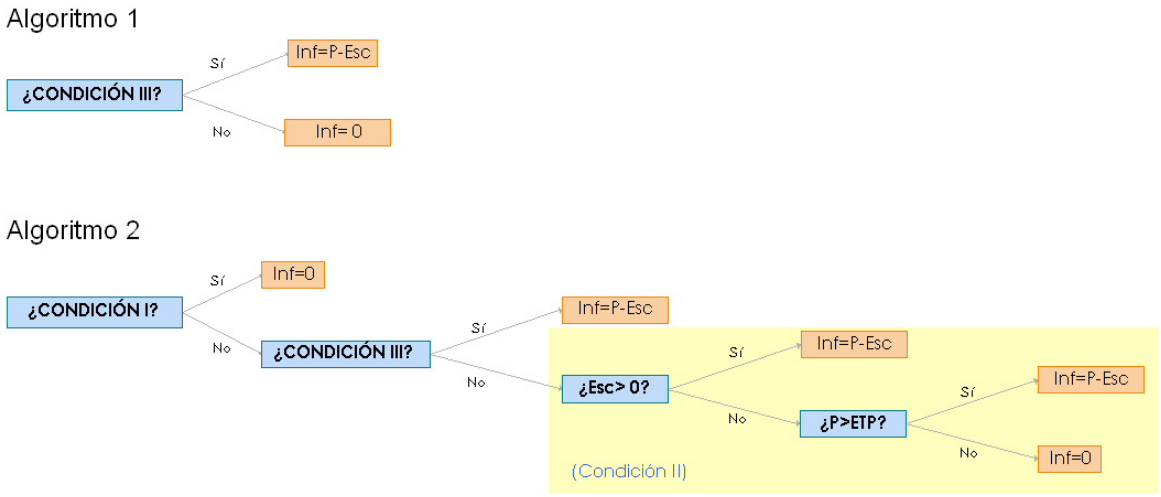

Figura E.2: Algoritmos de infiltración

Condición de humedad	Esc	P	Cálculo de Inf
Condición I	———	———	Inf = 0
Condición II	Esc>0	———	Inf = P-Esc
	Esc<=0	P>ETP	Inf = P-Esc
	Esc<=0	P<=ETP	Inf = 0
Condición III	———	———	Inf = P-Esc

Tabla E.3: Algoritmo 2 para calcular la infiltración

E.3. Cálculo práctico

Todos los cálculo que se detallan en el apartado E.2 han sido trasladados a la aplicación produciendo una serie de operaciones y comunicaciones con los diferentes WPS que se detallan a continuación:

E.3.1. Datos de entrada

Común a todo el proceso

- Tabla de número máximo de horas de luz por día en función del mes y la latitud
- Tabla de número de curva (NC) corregidos.

Para cada recinto

- Latitud
- NC del recinto
- Temperatura media mensual (12 medidas, 1 por mes)

- Serie diaria de precipitaciones

E.3.2. Datos de salida

- Para cada recinto
- Fecha de inicio
- Fecha de fin
- Serie diaria con:
  - Fecha
  - Precipitación
  - Escorrentía
  - Infiltración (por los dos algoritmos)

E.3.3. Cálculos

1. Calcular serie P5, suma móvil de las precipitaciones.
   a) Servicio**:** *PolynomialSolver*
   b) Cálculo**:** 5daysAccumulatedPrecipitation
   c) Datos de entrada: (día, P)*
   d) Datos de salida: (día, P5)*
   e) Fórmula: $P5 = \sum_{i-6}^{i-1} P$
2. Obtener codición humedad del suelo
   a) Servicio: *DecisionTableEval*
   b) Tabla de decisión: HumidityCondition
   c) Datos de entrada: (fecha, mes, P5)*
   d) Datos de salida: (fecha, condición) *
3. Obtener NC corregido
   a) Servicio: *DecisionTableEval*
   b) Tabla de decisión: CorrectedNC
   c) Datos de entrada: (NC, condiciónI), (NC, condiciónII), (NC, condiciónI-II)
   d) Datos de salida: (condicion,NCcorregido)*
4. Obtener infiltración máxima (S) en función del número de curva

a) Servicio: *PolynomialSolver*

b) Cálculo: MaxInfiltration

c) Datos de entrada: (NC-I, NC-II, NC-III)

d) Datos de salida: (NC-I, S) (NC-II, S) (NC-III, S)

e) Fórmula: $S = 254\left[\frac{100}{NC} - 1\right]$

5. Obtener serie de escorrentía

a) Servicio: *PolynomialSolver*

b) Cálculo: Runoff

c) Datos de entrada: (fecha, S, precipitacion)*

d) Datos de salida: (fecha, Esc)*

e) Fórmula: $\begin{cases} P \leq 0,2S & \rightarrow Esc = 0 \\ P > 0,2S & \rightarrow Esc = \frac{(P-0,2)^2}{(P+0,8S)} \end{cases}$

6. Obtener serie de infiltración por el método I

a) Servicio: *PolynomialSolver*

b) Cálculo: Infiltration1

c) Datos de entrada: (fecha, precipitación, Esc, condicionHumedad)*

d) Datos de salida: (fecha, Inf1)

e) Fórmula: $\begin{cases} si\ Condicion\ III & \rightarrow Inf = P - Esc \\ sino & \rightarrow Inf = 0 \end{cases}$

7. Obtener nº máx horas de sol mes

a) Servicio: *DecisionTableEval*

b) Tabla de decisión: MaxSunHours

c) Datos de entrada: (mes, latitud)*

d) Datos de salida: (mes, horas sol) *

8. Obtener ETP. (Pestaña ETP)

a) Servicio: *PolynomialSolver*

b) Cálculo: Thornthwaite_ETP

c) Datos de entrada: (mes, temperatura media, horas sol, nº días)*

d) Datos de salida: (mes, ETP)*

Desglose cálculo (dentro de *PolynomialSolver*):

1) Calcular el indice de calor mensual (i) para cada mes:

a' Datos de entrada: (mes, temperatura media)*

b' Datos de salida: (mes, i)*

c' Fórmula:$i = \left(\frac{t}{5}\right)^{1,514}$

2) Calcular índice de calor anual

a' Datos de entrada: (mes, i)*

b' Datos de salida: (I)

c' Fórmula: $I = \sum i$

3) Calcular la constante a

a' Datos de entrada: (I)

b' Datos de salida: (a)

c' Fórmula:$a = 675 \cdot 10^{-9} I^3 - 771 \cdot 10^{-7} I^2 + 1792 \cdot 10^{-5} I + 0,49239$

4) Calcular ETP sin corregir:

a' Datos de entrada: (a, (mes, temperatura media)*)

b' Datos de salida: (mes, ETPsin corregir)

c' Fórmula: $ETP_{sincorregir} = 16\left(\frac{10t}{I}\right)^a$

5) Obtener serie ETP corregida:

a' Datos de entrada: (mes, ETPsin corregir, num horas sol, días mes)*

b' Datos de salida: (mes, ETP)*

c' Fórmula: $ETP_{corregida} = ETP_{sincorregir} \frac{N}{12} \frac{d}{30}$

9. Obtener serie de infiltración por el método II

a) Servicio: *PolynomialSolver*

b) Cálculo: Infiltration2

c) Datos de entrada: (fecha, precipitación, Q, condicionHumedad, ETP)*

d) Datos de salida: (fecha, Inf2)*

e) Fórmula: $\begin{cases} si\ ((condicion = III)\ or\ (condicion = II\ and\ Esc > 0)) & \rightarrow Inf = P - Esc \\ \quad or\ (condicion = II\ and\ Esc <= 0\ and\ P > ETP) & \\ sino & \rightarrow Inf = 0 \end{cases}$

Anexo F

Definición y explicación de términos

En este anexo se van a definir y explicar términos de tecnologías o herramientas utilizadas durante el transcurso de este proyecto.

F.1. Servicios web

Los servicios web constituyen una tecnología emergente impulsada por el deseo de exponer de forma segura la lógica de negocios en Internet. Suponen el siguiente paso en la evolución de la tecnología orientada a objetos, y representan una revolucion al alejarse de las arquitecturas tradicionales tipo cliente-servidor a nuevas arquitecturas distribuidas tipo igual-a-igual (peer-to-peer).

Estos servicios consisten en un conjunto de estándares que permiten a los desarrolladores implementar aplicaciones distribuidas, utilizando herramientas muy distintas que facilitan que ciertos servicios de red sean dinámicamente descritos, publicados, descubiertos e invocados en un ambiente distribuido.

Uno de los estándares más utilizado para la creación de servicios web es XML, apareció en Febrero de 1998 y ha revolucionado la forma de estructurar, describir e intercambiar la información. Hoy en día, todas las tecnologías de servicios web se basan en XML. El diseño de este lenguaje se deriva de dos fuentes principales: SGML y de HTML. Entre los beneficios principales de los servicios web, cabe destacar:

- Los servicios web pueden ser accedidos desde cualquier plataforma, lo que contribuye a sistemas más flexibles y estables.
- Interoperabilidad dinámica, ya que los servicios web aseguran una interoperabilidad completa entre sistemas.
- Cualquier dispositivo que trabaje con HTTP y XML puede acceder a los servicios web.
- Los servicios web se basan en especificaciones estándar para el intercambio de datos, mensajería, búsqueda, descripción de la interfaz y coordinación de los procesos.

- Mayor agilidad y flexibilidad debido a una mejor integración con los sistemas existentes.

F.2. OGC y OpenGIS

OGC y OpenGIS OGC es una organización sin ánimo de lucro, fundada en 1994, dedicada a la promoción de nuevas aproximaciones técnicas y comerciales para geoprocesamiento abierto e interoperable. Este consorcio, impulsado por el impacto de Internet, se ha interesado por explotar las posibilidades ofrecidas por esta red. OGC ha desarrollado una serie de estándares OpenGIS que consisten en facilitar a a través de la web los intercambios de información entre la comunidad geográfica, definiendo los estándares que permiten acceder a los datos geoespaciales y asegurando a la interoperabilidad de los sistemas de información geográfica. OpenGIS define:

- Un Modelo de Geodatos Abierto: un conjunto general y común de información geográfica básica que puede ser usada para las necesidades de los geodatos o más aplicaciones específicas, utilizando métodos basados en objetos y métodos e e convencionales de programación.
- Servicios OpenGIS: el conjunto de servicios necesarios para acceder y procesar los tipos geográficos definidos en el Modelo de Geodatos Abierto y para ofrecer a capacidades para compartir geodatos dentro de las comunidades de usuarios quienes utilizan un conjunto común de definiciones geográficas y de traslado de éstas entre diferentes comunidades de usuarios que utilizan un conjunto e diferente de definiciones de características geográficas.
- Un Modelo de Comunidades de Información que emplee el Modelo de Geodatos o Abiertos y los Servicios OpenGIS en un esquema que establece:
  - Una forma efectiva y eficiente para una comunidad de productores de geodatos y usuarios que ya están compartiendo un conjunto común de definiciones de características geográficas para que eficiente y efectivamente mantengan estas definiciones y cataloguen y compartan estos conjunto de datos.
  - Una manera eficiente, exacta y óptima para que las diferentes comunidades de usuarios de geodatos y productores compartan geodatos en sus múltiples definiciones geográficas.

A continuación se van a describir el estándar referenciado en el proyecto, Web Processing Service (WPS).

F.3. Web Processing Service

Web Processing Services (WPS) define una interfaz estandarizada que facilita la publicación de procesos geoespaciales, el uso y el acceso a ellos por parte de

clientes. Por “proceso” se entiende cualquier algoritmo, cálculo que opere en datos espacialmente referenciados. Por “publicación” se entiende poner a disposición de los clientes (otras máquinas o personas) la información necesaria para la lectura de información y uso de servicios geoespaciales. Un WPS puede ser configurado para ofrecer cualquier tipo de funcionalidad SIG a un cliente a través de una red, incluyendo el acceso a cálculos preprogramados o modelos de computación que operen sobre datos espacialmente referenciados.

Los datos requeridos por el WPS pueden ser proporcionados a través de la red o pueden estar en el servidor. Esta especificación de interfaz proporciona mecanismos para identificar los datos espaciales requeridos para el cálculo, iniciar dicho cálculo y gestionar la salida para que el cliente pueda acceder a ella.

La especificación de los WPS está diseñada para permitir a un proveedor de servicios ofrecer un proceso vía Web (como por ejemplo intersección de polígonos), de manera que los clientes puedan introducir datos y ejecutar procesos sin tener conocimiento del API que lo permite. La interfaz WPS ofrece un estándar para normalizar el modo en que los procesos, sus entradas y sus salidas son descritas, cómo un cliente puede requerir la ejecución de un proceso y cómo la salida de dicho proceso es entregada al cliente. La interfaz WPS especifica tres operaciones que pueden ser requeridas por un cliente y llevadas a cabo por un servidor WPS. Todas son de obligado cumplimiento por parte de todos los servidores. Dichas operaciones son:

- GetCapabilities – Esta operación permite a un cliente solicitar y recibir metadatos de servicios (Capabilities) en documentos que describen las posibilidades ofrecidas por la implementación de un servidor dado. La operación GetCapabilities proporciona los nombres y descripciones generales de cada uno de los procesos ofrecidos por el servidor WPS.
- DescribeProcess – Esta operación permite a un cliente solicitar y recibir información detallada sobre los procesos que pueden ser ejecutados en el servidor, incluyendo los requerimientos de entrada, los formatos permitidos y las salidas que puede producir.
- Execute – Esta operación permite a un cliente ejecutar un proceso determinado implementado por el servidor WPS usando los valores de entrada proporcionados por el cliente y devolviendo la salida producida.

Estas operaciones tienen muchas similitudes con otros servicios Web del OGC, incluyendo WMS, WFS y WCS.

F.4. Web Map Service

El servicio Web Map Service (WMS) definido por el OGC (Open Geospatial Consortium) produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa"

como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Un mapa no consiste en los propios datos. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG, y opcionalmente como gráficos vectoriales en formato SVG (Scalable Vector Graphics) o WebCGM (Web Computer Graphics Metafile).

El estándar define tres operaciones:

1. Devolver metadatos del nivel de servicio.
2. Devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos.
3. Devolver información de características particulares mostradas en el mapa (opcionales).

Las operaciones WMS pueden ser invocadas usando un navegador estándar realizando peticiones en la forma de URLs (Uniform Resource Locators). El contenido de tales URLs depende de la operación solicitada. Concretamente, al solicitar un mapa, la URL indica qué información debe ser mostrada en el mapa, qué porción de la tierra debe dibujar, el sistema de coordenadas de referencia, y la anchura y la altura de la imagen de salida. Cuando dos o más mapas se producen con los mismos parámetros geográficos y tamaño de salida, los resultados se pueden solapar para producir un mapa compuesto. El uso de formatos de imagen que soportan fondos transparentes (e.g., GIF o PNG) permite que los mapas subyacentes sean visibles. Además, se puede solicitar mapas individuales de diversos servidores.

El servicio WMS permite así la creación de una red de servidores distribuidos de mapas, a partir de los cuales los clientes pueden construir mapas a medida. Las operaciones WMS también pueden ser invocadas usando clientes avanzados SIG, realizando igualmente peticiones en la forma de URLs. Existe software libre, como las aplicaciones GRASS, uDIG, gvSIG, Kosmo y otros, que permite este acceso avanzado a la información remota, añadiendo la ventaja de poder cruzarla con información local y disponer de una gran variedad de herramientas SIG.

F.5. Rich Internet Application

Rich Internet Applications (Aplicaciones Ricas de Internet) es un nuevo tipo de aplicación con más ventajas que las tradicionales aplicaciones web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales. Normalmente en las aplicaciones web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce a un tráfico muy alto entre el cliente y el servidor, llegando muchas veces a recargar la misma página con un mínimo cambio. Otra de las desventajas de las tradicionales aplicaciones web es la poca capacidad multimedia que posee. En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga

toda la aplicación y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos. Las capacidades multimedia son totales gracias a que estos entornos tienen reproductores internos y no hace falta ningún reproductor del Sistema Operativo del usuario.

F.6. Google Web Toolkit

Google Web Toolkit GWT es una herramienta, un *toolkit*, para la creación de aplicaciones web con AJAX mediante Java. A partir de un conjunto de clases Java, crea código HTML y JavaScript que podrá ser visualizado en diferentes navegadores web. De esta forma se permiten usar diferentes herramietnas utilizadas en el desarrollo de aplicaciones Java para el desarrollo de aplicaciones web. Asimismo se evita el uso de JavaScript y su falta de modularidad y dificultad de reutilización.

Las características de GWT son las siguietnes:

- Librería estándar de Java: permite usar un subconjunto de las clases de la librería estándar de Java.
- Widgets y elementos: posee un conjunto de elementos de interfaz gráfica de usuario listos para ser utilizados.
- Abstracción del navegador: ya que se está programando en Java con elementos de interfaz, widgets. No se requiere ningún conocimiento específico de HTML y las ligeras variaciones existentes entre los navegadores (Internet Explorer, Firefox, Safari, ...)
- JUnit: integracion con JUnit para realizar pruebas al código generado.
- Uso de JavaScript: aunque una de las principales razones para el uso de GWT es evitar usar JavaScript, permite integrar métodos JavaScript con el resto del código, útil para integrar librerías o elementos ya implementados.
- Internacionalización: tiene definido un mecanismo para el uso de un conjunto de clases dependiendo del idioma definido en tiempo de ejecución, lo que permite variar los mensajes e imágenes mostrados dependiendo del idioma escogido por el usuario de la aplicación.

Además, dado que se está trabajando con Java, se puede aprovechar el uso del entorno de desarrollo Eclipse para facilitar la implementación. Dentro de Eclipse podrá usarse su sistema de depuración, tarea muy compleja y laboriosa en aplicaciones JavaScript y en aplicaciones web en general, facilitada ahora con GWT.

Las comunicaciones con la parte servidor se realizan mediante AJAX, tarea que será abstraída por completo. Para el manejo de llamadas asíncronas se usará un sistema propio de GWT de llamadas remotas. En la parte cliente se define una interfaz con las llamadas que se deseen realizar al servidor. Esta interfaz será implementada desde la parte servidor, la cual podrá usar cualquier librería Java puesto

que el código de la parte servidor no será convertido a HTML, y Java Script como el de la parte cliente. Todos los parámetros (al igual que el valor devuelto) de las llamadas remotas tendrán que ser serializables, para serlo deben de cumplir ciertas restricciones (que se componga de tipos serializables). Mediante este mecanismo se permite la comunicación cliente/servidor intercambiando objetos. En la figura F.1 puede verse la estructura global explicada anteriormente.

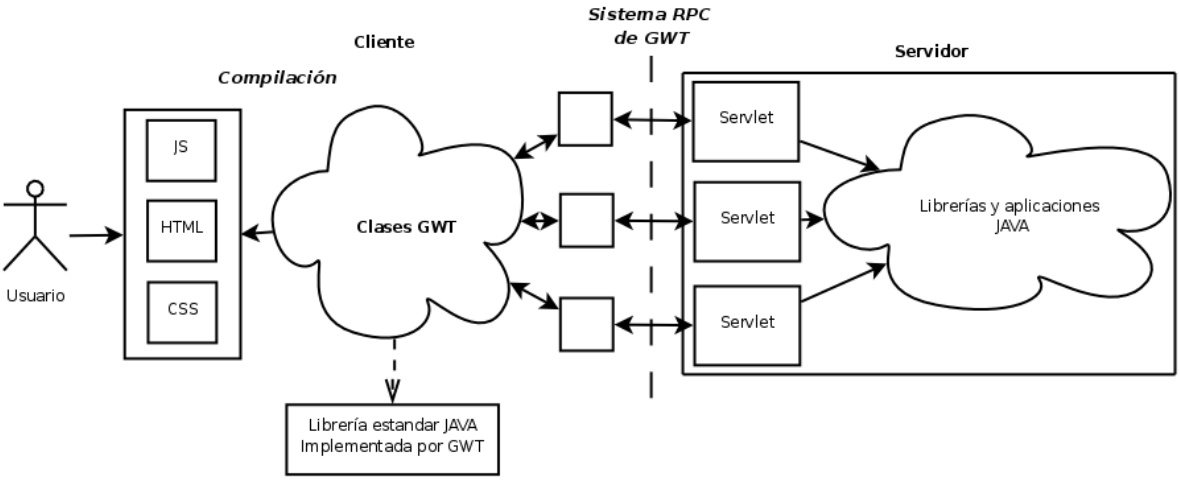

Figura F.1: EsquemaGWT

F.7. Maven

Maven es una herramienta para la gestión de proyectos software y la automatización de su construcción. Se basa en un formato XML, project object model (pom), para el uso y configuración.

Mediante la definición dada por el modelo *Project Object Model*, se puede especificar cómo se va a construir el proyecto, qué dependencias se requieren y qué acciones adicionales se necesitan realizar. El tratamiento de las dependencias permite tratar recursividad. Esto significa que cuando en un proyecto, que se quiere construir, se indica que depende de otro concreto, al incluir dicho proyecto en la construcción actual, se mirará a su vez el *pom* de ese proyecto para tratar de nuevo sus dependencias, de esta forma se consigue simplificar enormemente la tarea de uso de librerías externas. Otra de las ventajas significativas de Maven, es la utilización de repositorios para la obtención de las dependencias. Maven puede conectarse a éstos (existiendo un repositorio central y pudiendo definir y crear repositorios personales de forma sencilla) para la obtención de librerías.

Para el proceso de construcción, se sigue un "ciclo de vida" definido. Cada uno de estos objetivos require que se hayan cumplido los anteriores exitosamente. El ciclo de vida en Maven es el siguiente: *compile*, *test*, *package*, *install*, *deploy* (compilar, pruebas, empaquetar, instalar y desplegar respectivamente). En la ejecución de maven hay que indicarle qué objetivo se desea alcanzar (por ejemplo, *install*),

para lo cual Maven comprobará si los anteriores objetivos han sido cumplidos, si no fuera así se ejecutarían, y entonces es cuando el objetivo buscado es realizado (para poder ejecutar install, antes se debe de compilar, ejecutar las pruebas existentes y empaquetar los resultados).

Además del ciclo de vida básico definido en Maven, se puede alcanzar otros muchos objetivos mediante la utilización de *plugins*. Existen una gran cantidad de *plugins* que permiten realizar muchas tareas, como por ejemplo creación de documentación *javadoc*, informes de ejecución de pruebas, comprobación \emph{checkstyle} del formato del código fuente, creación de proyectos para integración con Eclipse, integración con GWT y muchos más. En este proyecto para la integración de GWT con Maven se ha usado el plugin maven-googlewebtoolkit[1].

A continuación se muestra un ejemplo de código \ref{codigo_maven} muy simple para ver el funcionamiento real de Maven. Aunque en este caso, es obvio que no sería ardua la tarea de construcción, en proyectos reales, el uso de Maven no es complejo, mientras que la construcción manual del proyecto sería una tarea laboriosa y difícil de llevar a cabo sin herramientas de automatización como Maven.

```
$ mkdir HolaMundo && cd HolaMundo
$ mkdir -p /src/main/java/com/company/hola # com.company.hola será el package
$ cat > /src/main/java/com/company/hola/Hola.java << EOF
package com.company.hola; public class Hola {

    public static void main(String [] args) {
        System.out.println("¡Hola mundo!");
    }

}
EOF
$ cat > pom.xml << EOF
<project>

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.company</groupId> # grupo al cual pertenece el proyecto
    <artifactId>HolaMundo</artifactId> # el nombre
    <version>1</version> # versión del componente

</project>
EOF
$ mvn install # install!, compila y genera el .jar
$ java -cp target/HolaMundo-1.jar
           com.company.hola.Hola # package.clase ¡Hola mundo!
```

F.8. BIRT

Es un plugin para Eclipse diseñado para la generación de informes en aplicaciones web. Aspectos a tener en cuenta son los siguientes:

[1] http://code.google.com/p/gwt-maven/

- Proyecto de fundación Eclipse, con licencia Eclipse Public License, menos restrictiva que la LGPL.
- Pensado para tratar fuentes heterogéneas, JDBC, XML ...
- El formato de informes (en XML) tiene una maquetación tabular, orientado a flujo. Soporta aplicar estilos mediante uso de hojas de estilo CSS.
- Tiene un visor web para visualizar informes y exportar a otros formatos.
- El interfaz gráfico se basa en Eclipse, muy potente y con mucha ayuda, por lo que es relativamente sencillo de usar.
- Bastante documentación, pero más centrada a usar BIRT como herramienta final que para su integración
- BIRT está formado por dos componentes: Birt designeer con el que se pueden diseñar los informes desde el propio Eclipse y un generador de informes que puede ser integrado en un entorno Java.

Índice de figuras

Índice de tablas

Bibliografía

[1] Chaganti, P. *GWT Java Ajax Programming.* Packt Publising Ltd., 2007.

[2] Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns.* Addison Wesley, 1995.

[3] Murrel, P. *R Graphics.* Chapman & Hall / CRC, 2005

[4] Weathersby, J., French, D., Bondur, T., Chatalbasheva, I. *Integrating and Extending BIRT, 2nd Edition.* Addison-Wesley, 2008

[5] Official BIRT Documentation *BIRT design tutorial* http://download.eclipse.org/birt/downloads/examples/reports/2.1/tutorial/tutorial.html

[6] Rosenbaum, S., Weathersby, J. Oda extensions and *BIRT.* Eclipse Magazine (2007)

[7] Sperberg-McQueen, C. M., Thompson, H. *W3C XML Schema.* http://www.w3c.org/XML/Schema.

[8] Open GIS Consortium Inc. *Web Map Service (WMS).* http://www.opengeospatial.org/standards/wms

[9] Open GIS Consortium Inc. *Web Processing Service (WPS).* http://www.opengeospatial.org/standards/wps